USING INDUCTIVE INFERENCE OF
PAST PERFORMANCE TO BUILD STRATEGIC
COGNITIVE ADVERSARY MODELS

BY

STEVEN MICHAEL WALCZAK

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1990

ACKNOWLEDGMENTS

a great debt of gratitude to my committee chairman, Paul Fishwick. He spent many tireless hours reviewing and analyzing my research and encouraging me.

My greatest human source of aid, comfort, and encouragement during my graduate studies has been my wife. Karen gave three years of her life to work and get us both through graduate school. She has my undying love.

I thank God for giving me the wisdom, intelligence, and motivation to pursue my doctoral degree. The knowledge of His presence in my life has been the single most important factor in my ability to complete graduate school. I hope and pray that I will be faithful and wise enough to hear His calling in my life, that I may choose to follow His path.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ix

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

USING INDUCTIVE INFERENCE OF
PAST PERFORMANCE TO BUILD STRATEGIC
COGNITIVE ADVERSARY MODELS

By

Steven Michael Walczak

December 1990

Chairman: Paul A. Fishwick
Major Department: Computer and Information Sciences

To perform optimally in adversarial domains, artificial intelligence programs
must be able to identically evaluate domain situations and develop plans as the
adversary. Knowledge about an adversary's evaluation criteria and objectives is
difficult to obtain since an adversary will guard this information in order to promote
a favorable outcome for himself. We present an adversary modeling methodology
which is capable of acquiring the evaluation criteria of an adversary through
observations of the adversary's past performances in the domain.

Our adversary modeling methodology uses induction to capture perceptual
chunks that are repeatedly displayed by a specific adversary. The acquired chunks
are then used to predict the adversary's actions during future encounters. Pattern
recognition is used to identify perceptual chunks in the current adversarial domain

situation which are analogous to chunks that have been previously learned from the current adversary.

We have implemented our adversary modeling methodology in the program IAM, Inductive Adversary Modeler, which has been applied to the domain of chess. When high quality knowledge is available about an adversary's past performances, the adversary modeling methodology accurately predicts ten percent of an adversary's future action plans. The ability to accurately predict an adversary's move choices reduces the complexity of the game tree search which enhances the playing ability of current chess programs.

CHAPTER 1
INTRODUCTION

Adversarial situations surround us in varied forms on a daily basis. Examples of adversarial situations include vying for a promotion at our place of employment, Monday night's football game, and military conflicts throughout the world. These adversarial situations are characterized by a minimum of two actors each of whom is attempting to realize a goal that conflicts with the goal of the opposing actor. The actors can be individuals or groups of individuals such as a football team players that are cooperating to achieve the goal of the actor.

When humans are faced with the problem solving task of realizing a goal against the desires of an adversary, they employ a variety of techniques that enable them to outperform their adversary and accomplish their goal. The standard cognitive problem solving methods of difference reduction and working backwards are insufficient in adversarial domains. Human competitors use deductive reasoning and analogy to previous domain situations to realize their goals in adversarial domains. The use of analogy to gain a strategic advantage over an adversary necessitates the studying of a specific adversary's previous performance, preferably in situations analogous to the upcoming contest.

Human competitors performing in adversarial domains study the past performances of their adversaries to identify strengths and weaknesses which can be

exploited respectively to gain a strategic or tactical advantage. This information is preserved by the competitor as a model of the likely actions an adversary will perform in particular situations. The model of the adversary's probable actions is then exploited when an appropriate situation occurs during the course of the competition. Such situations arise when the current game state--situation in the adversarial domain--is analogous to a game state which occurred in the adversary's past.

1.1 Problem Statement

Current artificial intelligence application efforts in adversarial domains are not performed optimally. A factor which keeps adversary-based programs from performing at expert levels is the lack of knowledge about the potential adversary. Lenat and Feigenbaum (1987) claim that domain specific knowledge is required to solve difficult problems and that the performance of intelligent programs improves as additional domain knowledge is utilized.

Acquiring knowledge for programs in adversarial domains is hindered by the fact that adversaries are unwilling to reveal their tactical and strategic goals, since this would be counter-productive to the attainment of their goals. Candland (1980) and Bolc (1987) have claimed that knowledge about any domain is available from a wide variety of sources. Lenat (1982) proposes the use of the observable actions of the adversary while performing in the adversarial domain. The proper selection of adversarial domain situations to observe guarantees the acquisition of high quality knowledge.

For the domain of chess, lack of knowledge about an adversary's tactics and strategies is analogous to searching the game tree to a depth of only one ply. All moves are considered as winning moves and the game proceeds in a random manner. To acquire knowledge for use in adversarial domain application programs, we will emulate the traits of human experts performing in adversarial domains, by analyzing the previous performances of a particular adversary in relevant domain situations.

1.2 Contribution of the Dissertation

Adversarial domain programs require knowledge about an adversary's tactics and strategies to continue to improve their performance. We need to consider two separate factors that affect the strategic decisions an adversary will make. An adversary's strategic decisions are affected by the plans or scripts that the adversary feels will satisfy the current strategic goal. Before the adversary can select a plan of action however, the domain situation must be evaluated to determine which plans are appropriate to that particular situation.

Current programs performing tactical and strategic planning for adversarial domains rely heavily on tree structures and search techniques such as iterative deepening and alpha-beta pruning. The ability to accurately predict the choices that will be made by an adversary in specific domain situations reduces computational complexity and increases the depth of search. The increased depth of search that our research enables is directly responsible for the augmented tactical ability of these search based programs.

The purpose of our research is to develop a generalized method for modeling the strategic decision making process of an adversary. The information flow diagram for our adversary modeling methodology is shown in Figure 1. Our adversary modeling methodology makes use of knowledge which is currently not used by programs operating in adversarial domains. By increasing the knowledge available to tactical and strategic planners, we will improve the overall performance of these adversarial domain applications.
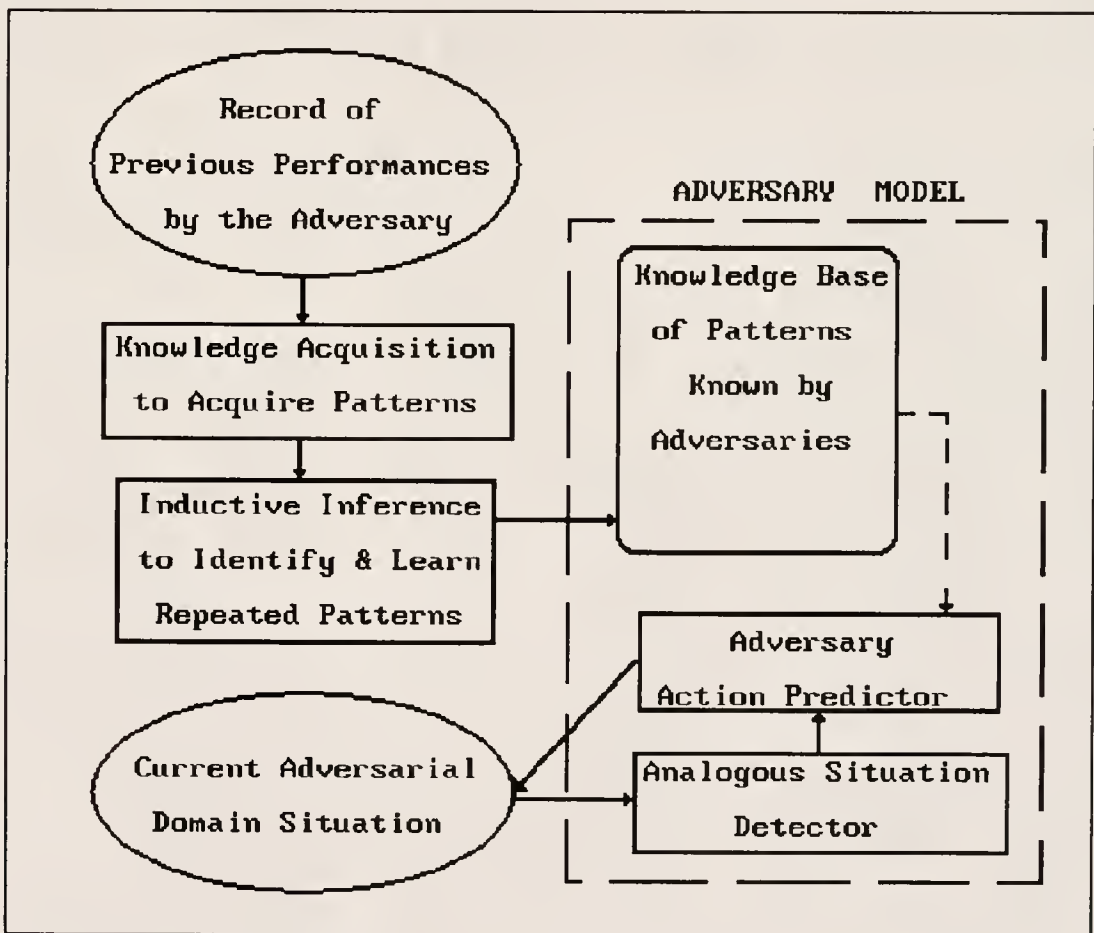
Figure 1: Logic flow diagram.

Our adversary modeling methodology uses psychological principles of organization and evaluation such as the Gestalt principles of organization (Anderson,

1980, and Pomerantz, 1986) to emulate the primary evaluation criteria used by an adversary. Kodratoff and Michalski (1990) note that recent machine learning efforts have focused on building cognitive learning architectures. We intend for the adversary modeling methodology to be applicable across all adversarial domains and hence, we include as little domain specific knowledge as possible. This means that we are content to acquire the primitive elements that an adversary uses in evaluating a domain situation without embellishing those elements with further domain specific knowledge.

The psychological principles of organization are applied to a collection of previous adversary performances in the domain (e.g., chess games played by the adversary in prior tournaments and matches). We acquire the cognitive chunks used by an adversary to evaluate the current domain situation. The acquired cognitive chunks of the adversary help us to view the domain the same as our adversary and to avoid influencing domain evaluations with personal biases.

We now have a large collection of chunks that have been displayed by the adversary. The chunks are either textual patterns corresponding to the adversary's verbal memory or geometric patterns corresponding to the adversary's visual memory. The chunks we have acquired contain patterns that have been used only once and may have occurred by chance. We use induction to eliminate these chance chunks and create a valid and useful adversary model. The inductive phase identifies chunks which are repeated in two or more games. The repetition of patterns is used to indicate the probability that an adversary will use the acquired chunk in future

evaluations of domain situations. Because we are using induction as our learning method, the adversary model created will be used as a heuristic guide instead of an algorithmic solution.

Following the inductive learning, the collection of chunks that have been repeatedly displayed by the adversary are stored in the adversary model. We consider these chunks to be the primitive elements used by an adversary in evaluating current domain situations. The chunks of the adversary model are then used to predict adversary decisions when analogous situations occur in the adversarial domain.

The knowledge obtained by the adversary modeling methodology is highly reliable, because it comes directly from the adversary. A certain level of care must be taken to guarantee that the adversary is performing normally in the domain and not providing misinformation to the modeling methodology. We verify the adversary modeling methodology by implementing the methodology for the adversarial domain of chess. The implementation is called IAM, for Inductive Adversary Modeler. Our justification for selecting the chess domain is given in section 2.10.6.

Berliner and Ebeling (1988) note that speed is a critical factor for evaluation functions in chess as well as other adversarial domains. The knowledge that we are learning about an adversary is available as historical records prior to the current performance of the adversarial domain program. Therefore, the adversary modeling methodology's knowledge acquisition can be performed prior to its use by the domain program. This preprocessing effectively eliminates any time or complexity costs

generated by the adversary modeling methodology from being added to the current program's evaluation function cost (Aho, 1974). The adversary model acts as a heuristic coach to an existing domain program by predicting which moves an adversary will make, as shown in Figure 2. The only part of the adversary model that impacts the time order of the existing evaluation function is the analogous situation detector that was shown in Figure 1.
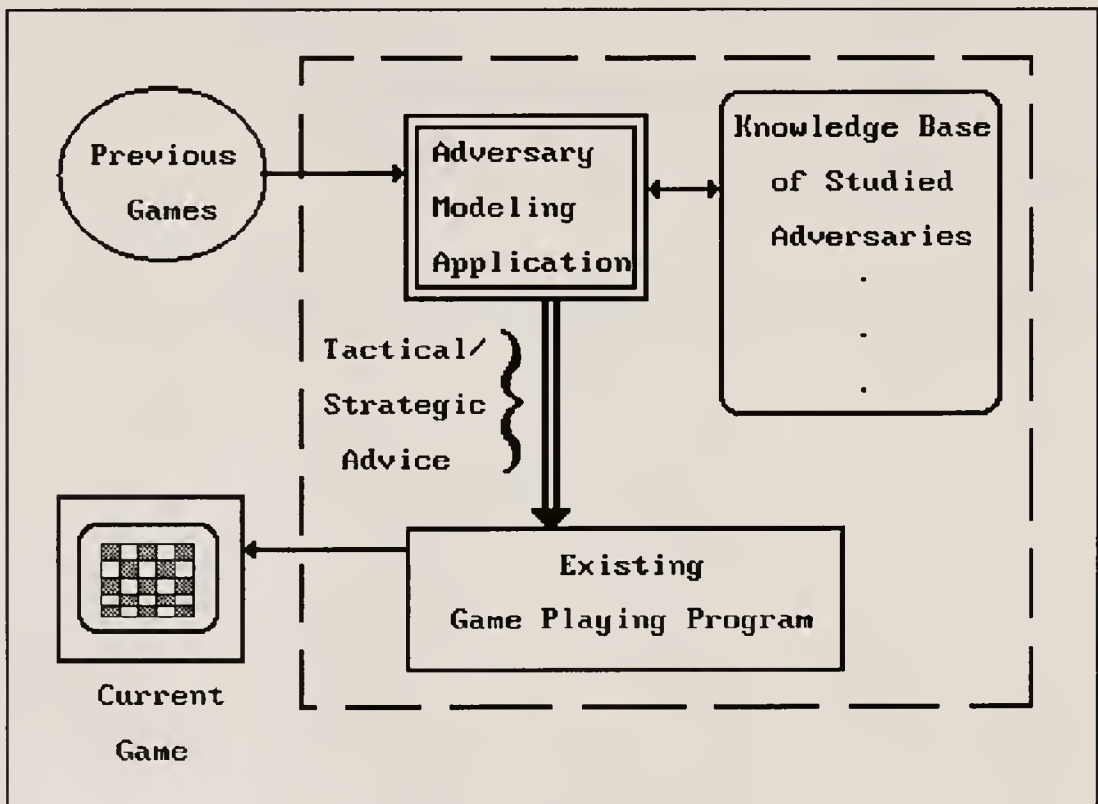
Figure 2: Overview of the physical system.

1.3 Outline of the Dissertation

We have presented a general overview of the problem of the lack of knowledge about adversaries available to intelligent programs operating in

adversarial domains. Our solution to this problem has been briefly presented in this chapter.

In Chapter 2, we present background material on induction and psychology that affects our research. The research that has already been performed in several of the adversarial domains including the domain of chess is discussed. Finally, we present the reasons for selecting the domain of chess as our application domain.

In Chapter 3, we detail our research by first discussing the implementation method and then the operation of the adversary modeling methodology. Each of the acquisition and induction steps that have been briefly described in this chapter are presented in detail.

Chapter 4 contains an example of the adversary modeling methodology in action and provides quantitative performance results of our application program, IAM. Our results and their impact on artificial intelligence search methods are provided in Chapter 5. We discuss our conclusions and future research directions in Chapter 6. The Appendix contains a glossary of chess and artificial intelligence terms to aid the reader in understanding some of the domain descriptions that we have presented.

CHAPTER 2
BACKGROUND AND RELATED WORK

In this chapter, we review the theoretical and applied work on induction. We also describe the theoretical work on game tree search to emphasize our contribution to this research area. Finally, we describe related work in the various adversarial domains that have led to our solution with a detailed review of the domain of chess which serves as our application domain. For each of the adversarial domains presented, we describe the use of adversary modeling by humans and any applied research that affects the implementation of our adversary modeling methodology.

2.1 Induction

Mankind has been using induction to perform problem solving and learning tasks for many millennia. Rosch (1973) describes the way humans form concepts and categories about their natural world. Polya (1973), in the early 1950s, formalized the concept of induction. Induction is the process of discovering general laws or concepts from observations and combinations of particular instances.

An example of induction for a natural world concept would be the generalization that all ravens are black. This generalization is based on the observation of a limited number of instances of a raven. Without seeing all the possible examples or instances of raven, we are confident to induce that the color of

a raven is the same as the color of the instances we have seen--black. The strength of our belief increases as more and more examples of black ravens are observed. However, only one instance of a nonblack raven needs to be identified to invalidate our induction.

From the above example we can see that inductive inference is learning in the limit (Daley & Smith, 1986). We need to see sufficient observations of the right type to form a correct generalized concept. Since we cannot guarantee that the right type of observations are being provided to an inductive inference system, we must rely on the use of a large number of examples of category instances to acquire the desired concept. This means that the adversary modeling methodology we have developed should perform more accurately as we provide greater numbers of examples of an adversary's domain performance.

Since Polya's formalization of the concept of induction, theoretical research on induction has primarily focused on the formation of concepts or classifications. Angluin and Smith (1983) and Case and Smith (1983) present rigorous discussions of the mathematical principles of induction. This theoretical background provides a solid base for the use of inductive inference as a computational tool for learning classes of objects or ideas.

The extraction of general principles from specific examples is a major research goal in machine learning (Anderson, 1987). Induction has been applied to many domains. Examples of applied inductive inference algorithms are Michalski's INDUCE and Quinlan's ID3 (Michalski et al., 1983 and 1986). Each of these

example applications attempts to classify instances of a domain problem (e.g., soybean diseases for the INDUCE program) into maximally-specific groups that will identify all known instances as well as accounting for unknown instances.

Prior to the theoretical discussions of inductive inference mentioned above, Blum and Blum (1975) had already demonstrated that the ability to play games, or at least to make technically correct moves, can be learned through inductive inference. Induction has not been utilized in game domains due to the lack of high quality play produced by Blums' algorithm. Our adversary modeling methodology focuses on a narrower slice of the game domains, namely predicting adversary move choices and playing style, and is able to learn high quality knowledge.

Adversarial domains present several specialized problems to the standard inductive inference algorithms like ID3. Langley (1985) has noted that feedback is an essential element of most learning methods. Inductive inference algorithms use feedback to verify acquired concepts and to identify concepts that need to be modified. In adversarial domains, feedback on the performance of an algorithm is a not attainable until after the current domain situation has ended. Because of the inherent time delay for feedback in adversarial domains, we substitute probabilistic reasoning so that our confidence in the current adversary model increases as additional examples of the same type are observed.

Another problem in adversarial domains is the dependence of the domain on time. Actions occurring in the domain are temporally related. The inductive inference applications cited above belong to the class of hierarchical or structure

induction methods. Structure induction methods are used to produce classification systems. Muggleton (1990) proposes an inductive method called sequence induction which has a time effect relationship to the domain. Sequence induction uses a series of domain descriptions that are altered by actions operating on the domain to produce control systems.

Mitchell et al. (1986) refer to a process of mechanized induction which receives domain events as input and produces concepts for predicting future events. For adversarial domains, sequence induction provides a method to produce evaluation criteria for predicting an adversary's strategic decisions.

2.2 Tree Search

Programs in the adversarial domains of games invariably use trees as their knowledge representation scheme. Finding solutions or move choices in game trees involves searching the tree for the optimal finishing value. Von Neumann and Morgenstern (1953) published the Minimax algorithm for backing the final game values up through the game tree to the current node. Minimaxing attempts to account for the effects of an adversary in the game domain by alternately raising the minimum possible value or maximum possible value at each level of the tree.

If an exhaustive search can reach all of the leaf nodes of the tree, then the true finishing value of the game can be raised to the current node and an optimal game will ensue. However, games like chess, Go, and Othello have tree representations that hold 10^{40} nodes or more. Game trees of this size prohibit an

exhaustive search. Instead, game programs will search down to a preselected depth and apply an evaluation function that will determine a best guess for the true finishing value obtainable from that node.

Evaluation functions use domain knowledge to predict the outcome of the game from a specific position. All domain specific knowledge is contained in the evaluation function. The quality of knowledge has a direct effect on the playing ability of the game program (Berliner & Goetsch, 1984, and Lee & Mahajan, 1988). Northwestern University's CHESS4.6 was able to defeat other chess programs which searched several ply (levels) deeper into the game tree based on the strength of its evaluation function (Elithorn & Banerji, 1984, and Wilkins, 1980).

The primary problem faced by search algorithms using tree structures is the horizon effect. The horizon effect occurs when the solution to a search problem lies at a depth in the tree that is beyond the current search depth. This is a realistic problem for complex adversarial domains. In chess, the KRKN endgame, king and rook versus king and knight, can require a twenty-seven move solution or a game tree search depth of fifty-two ply (Bratko & Michie, 1980). Similarly, the KBBKN endgame has solutions of sixty-six moves or one hundred thirty-two ply search depth (Roycroft, 1988). Current chess programs search between eight and fourteen ply before selecting a move.

Two different schools of thought exist concerning methods for continuing to improve game program performance. The first school believes that the depth of the search is the sole criterion affecting the playing ability of game programs.

Adelson-Velsky et al. (1988) state that increasing the depth of search of a game tree increases the quality and reliability of game play. Berliner and Ebeling (1988) specifically state that a deep tree search will outperform a shallower search which uses more knowledge. Various techniques are used to increase the depth of search for a program. Limiting the breadth of search and pruning of undesirable branches through alpha-beta pruning and iterative deepening permit a deeper search for a static number of evaluated tree nodes (Barr & Feigenbaum, 1981). Specialized hardware can also be used to increase the speed and thereby the depth of search (Ebeling, 1986).

An example of alpha-beta pruning is shown in Figure 3. The four leaf nodes which have their branches cut by a diagonal line can each be pruned using alpha-beta pruning. Once the three value is elevated to the left subtree level two node, then the level one node is guaranteed a minimum value of three. Since each of the first leaf nodes below the center and right subtrees have lower values than three, the remaining leaf nodes do not need to be evaluated. This is called an alpha cutoff. A beta cutoff is similar except that the cutoff is performed based on a guaranteed maximum value for a Min node.

The other method for improving the performance of game programs is to increase the knowledge that is available to them. We have already seen that better quality knowledge enabled CHESS4.6 to outperform other programs that searched several ply deeper into the game tree. The use of additional knowledge has the same effect as searching deeper into the game tree. The projection ability of certain types
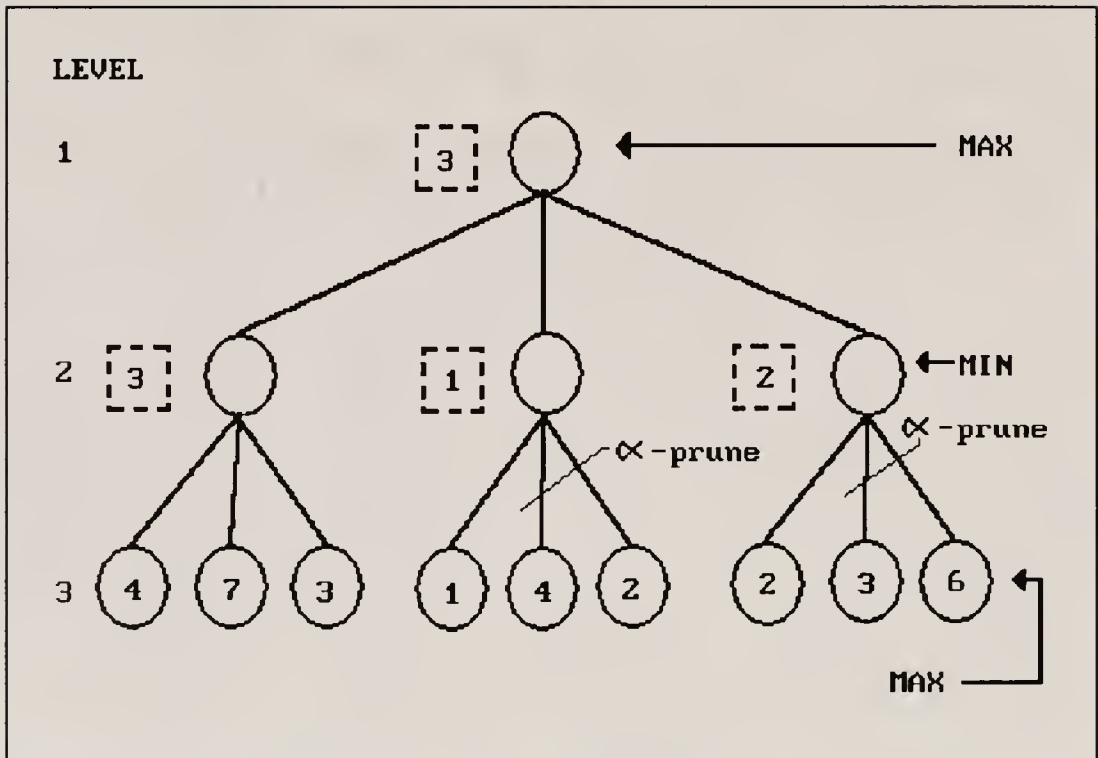
Figure 3: Sample game tree with alpha-beta pruning.

of knowledge in the chess domain is nine ply for tactical knowledge, twenty-five ply

for positional knowledge, and forty-five ply for strategic knowledge (in Elithorn &

Banerji, 1984). Current game programs use mostly tactical knowledge along with

some positional knowledge.

For adversarial domains, the use of domain knowledge can be dangerous by

leading us into a false sense of security. Samuel (1959) warns that programs using

the Minimax algorithm must take into consideration the intent of the adversary.

Cecchini (in Crookal et al., 1987) claims that the actual intent of the adversary will

require programs to use different heuristics in order to win. Human tree search is

goal directed and the direction and method of search may change as new information

becomes available (Frey, 1983). This last statement means that the preceding events in a game and tournament affect the future strategic decisions of the adversary.

Our adversary modeling methodology follows the school of thought which believes that more knowledge is required to continue improving the performance of game programs. To avoid the complications of adversarial intent, our heuristics are based on the primitive psychological elements that are used by an adversary to evaluate the domain. Application of the adversary modeling heuristics occurs when the current game board pattern is analogous to a previously induced situation. The use of pattern recognition to increase search depths has already been implemented by Samuel (1959 and 1967) in the form of signature tables and in the Russian chess program KAISSA (Adelson-Velsky et al., 1975) to perform cutoffs or pruning.

The theoretical increase in search depth which we have achieved through the adversary modeling methodology is detailed in Chapter 5. Several authors (Findler & Meltzer, 1971, and Utgoff, 1986) have noted that the economy of a search heuristic can be measured by the reduction in search effort and complexity which produces a deeper search.

2.3 Athletic Adversarial Domains

The modeling of an adversary by humans is well documented in athletic domains. Coaches of baseball (Aliston & Weiskopf, 1984), football (Riggins & Winter, 1984), and other sports use scouting reports and films of their opponents to study their opponent's strategies. This enables the coaches of athletic teams or

individuals to develop counter-strategies to foil the goals of the adversary and promote their own goals.

In a personal communication, Owen J. Holyoak of the Exercise and Sport Sciences Department at the University of Florida stated that the amount of preparation performed by athletic team coaches corresponds directly to the perceived threat of the adversary. When the technique of studying an adversary's past performance is not fully used, surprising results can occur. At the 1982 homecoming football game for the University of Florida (UF), the opponent was West Texas State University (WTS). Although WTS was considered to be an underdog and was not expected to score any points, they obtained a fourteen point lead early in the game by using some plays that UF had not anticipated. The lack of preparation on UF's part was responsible for the surprising start of the 1982 homecoming game.

2.4 Education Domain

The artificial intelligence domain of intelligent tutoring systems (ITSs) or intelligent computer-aided instruction has already made successful use of models of human thinking. These models are based on the availability of knowledge to the human and that human's mental set at a particular point in time. The people being modeled in ITS programs are the students who are attempting to learn. The student models are used to predict the sources of error in student thinking which cause incorrect responses to test items produced by the ITS.

Woolf (1984) has demonstrated an ITS which uses a model of the probable knowledge currently held by the student for discourse planning. This model adapts its understanding of the student's knowledge during the lesson to capture changes in the student's cognitive state precipitated by the acquisition of new knowledge. Our adversary modeling methodology operates similarly by constructing a model of the probable evaluation mechanisms known to an adversary to predict future strategic decisions of the adversary. Other examples of this modeling approach as used in ITSs can be found in Sleeman and Brown (1982).

2.5 Adversarial Game Domains

Because of their finite size, games provide us with a convenient abstraction of more difficult real world problems. The former Japanese use of the game of Go to train their military leaders (Reiss Games, 1974) demonstrates the practical extension of a game domain to a military domain. Pritchard (1973) elaborates further by claiming that Go is guerilla warfare on a grand scale.

We will be using two person zero-sum perfect information games (Zagare, 1984) to demonstrate the adversary modeling methodology. Zero-sum games are always adversarial in nature, since the wins of one side are equivalent to the losses of the other side. Our selection of perfect information games was made to permit an algorithmic evaluation of the results. The domain of chess which we will use for our application of the adversary modeling methodology is reviewed in section 2.10.

Other types of games, such as games of chance, that do not fit our selection of zero-sum perfect information games can also benefit from the adversary modeling methodology. Our contribution which reduces the complexity of search is usable by any game program that utilizes game tree search and heuristics to solve domain problems. Berliner's BKG backgammon program (in Levy, 1988) is an example of a game of chance which uses heuristic methods and game tree search to select between moves.

Christensen and Korf (1986) claim that heuristic methods for one person puzzles are equivalent to heuristic methods for two person games. The adversary modeling methodology is capable of predicting moves in one person games that will reduce the complexity of the current game position to one which has a known solution. Other research using induction of previous games has been performed for one person games. PREFER (Utgoff & Saxena, 1987) solves the 8-puzzle by performing induction on previous 8-puzzle solutions.

Samuel (1959 and 1967) was the first researcher to use machine learning to create an intelligent game playing program. Samuel's checker playing program uses rote learning techniques to learn how to play checkers. This program is capable of imitating the style of an adversary by mimicking the winning moves of the adversary presented in the set of training instances. Programs which rely on rote learning tend to become inflexible and require an equally large number of retraining instances to adapt to a different style of play. The method of mimicking an adversary has been shown to fail for the game of Go (Pritchard, 1973) and should not be considered as a generalized method of strategic planning against adversaries.

Samuel developed signature tables to reduce search complexity. Signature tables are used to remember the evaluation function value for specific positions. When identical positions are identified through a pattern matching mechanism, the signature tables are used to retrieve the corresponding node value. Wilcox (1985) has also applied pattern recognition techniques to the game of Go to construct NEMESIS, a strong Go playing program.

IAGO (Rosenbloom, 1982) uses the standard search techniques of alpha-beta pruning and iterative deepening to play championship-level Othello. Rosenbloom points out that the heuristics for IAGO need to change during different phases of the game to capture shifts in the strategic importance of various game principles. For example, in the middle game of Othello mobility is a critical factor, but in the end game portion of Othello the quantity of stable pieces is the primary concern. Berliner (1979) notes similar shifts in strategic reasoning for the domain of chess. Our adversary modeling methodology compensates for the strategic shifts at various points in the game by assigning time durations to the domain evaluation chunks that are acquired for each adversary.

GINA (DeJong & Schultz, 1988) tries to learn to play Othello by using experience gained from previous games. This approach is very similar to the one used by Samuel for his checkers program. GINA attempts to exploit the weaknesses of a specific opponent. Several of GINA's implementation features are applicable across adversarial domains. These features are:

- Serve as an external coach to an existing program in the domain.

- Only use observable behavior to learn about an adversary. For GINA, the observable behavior is the moves made by an adversary in prior games.

- Do not use feedback to modify learning, except for the results of the games that have been studied.

The best example of strategic planning based on an adversary's modus operandi is Waterman's (1970) poker player. Waterman has implemented a draw poker playing program which analyzes an opponent's style of play to decide when a bet is appropriate. This program learns over a period of several hands which heuristics to use against a particular adversary. We expand upon Waterman's idea by not limiting the program to a static number of predefined heuristics.

2.6 Law Enforcement Domains

Police departments and the Federal Bureau of Investigation each make use of adversary modeling. As an example, a police detective will try to predict when, where, and to whom a serial killer will perform his next crime. This is done primarily by analyzing the criminal's previous performances of crimes. Similar techniques are being considered for predicting the outcome of possible terrorist activities in the RISK project sponsored by Lawrence Livermore National Laboratories.

The key to successful drug enforcement is intelligence gathering and information processing (Witkin, 1990). Raw data is voluminous and must be converted to a useable form. Our adversary modeling methodology is capable of finding patterns in raw data that correspond to resource movement problems (e.g., the actual drugs and the monetary compensations) and can be used by law enforcement officials to increase their effectiveness.

2.7 Military Domains

Devastating results occur on the battlefield when sufficient preparation against an adversary is not performed. Due to different sociological, geographical, and educational backgrounds, potential adversaries will have unique behavioral rules (Bond, 1986 and Boring, 1945). These behavioral rules govern varying beliefs and goals concerning war and the use of nuclear weapons (McMillen, 1984 and Sokolovskiy, 1968). Dunnigan (1982) and Ermarth (1978) state that if we assume that an enemy will act strictly according to our own beliefs, then we will produce a tactical blindness. Tactical blindness produces the inability to predict or fathom the plans of our adversaries. Historic examples of the tactical and strategic blindness produced by conscripting our own beliefs onto an adversary are the bombing of Pearl Harbor by the Japanese in 1941, the defeat of the United States tactical forces at the Chosin Reservoir during the Korean War (Wallace, 1990), and the Tet Offensive of the North Vietnamese in 1968. Many soldier's lives might have been saved early in World War II if the United States soldiers had understood the Japanese attitude toward surrender (Boring, 1945).

2.7.1 Human Adversary Modeling Efforts in Military Domains

Military leaders (Gruner, 1988, and Ryan, 1988) claim that victory is often contingent upon knowing your enemy. This means that military planners must be able to evaluate the current situation using the same criteria as their adversary. A vital component of intelligence information is briefings on an adversary's operational characteristics or tactics (Gruner, 1988). Robert E. Lee, who is often cited as the greatest military strategist (B. Davis, 1956), is a practical example of the power accompanying detailed knowledge about an adversary. Lee was educated at West Point, where all of the Union commanders were also trained, and served along side many of the adversaries he would face in the Civil War during his tenure as an officer in the United States Army and Cavalry (Snow, 1867). Lee's detailed knowledge of the training and tactics of his adversaries enabled him to frequently outperform better equipped and larger forces.

The United States military makes use of this concept in training our soldiers. Each service branch maintains a cadre of personnel who are trained in Soviet-style war tactics and strategy (Robbins, 1988). The training centers for the Army, Air Force, and Navy are the National Training Center, Red Flag, and the Top Gun school respectively. The purpose of these Soviet analogous fighting forces is to provide a realistic training environment to better prepare the military for the possibility of an actual war against Soviet forces or forces that use Soviet-style tactics.

A historic example of adversary modeling for military purposes occurred during World War II. A detailed psychological profile of Hitler was constructed by

Langer (Langer, 1972) for the Allied forces. This model of Hitler was constructed solely from second-hand information. The psychological model of Hitler accurately predicted the method of his ultimate demise via suicide.

2.7.2 Computer Planning in Military Domains

Military games and strategic and tactical planners make use of varying levels of adversary modeling, ranging from none in typical multi-player games and training simulators to moderately advanced models in certain strategic planning programs. Failing to account for the actions of an adversary severely inhibits strategic planning. Military planning tools must account for the probable actions of an adversary based on the adversary's intentions and strategic style, otherwise tactical blindness will result.

The tactical troop movement planner TACPLAN (Andriole et al., 1986) is typical of most military planning tools. TACPLAN only accounts for static elements of the domain such as the effect of terrain on equipment movement. Knowledge about adversaries is nonexistent. The next step towards an adversary model is programs similar to ARES (Young & Lehner, 1986) which acknowledges the presence of an adversary. ARES is also a tactical planner; however, the plans generated by ARES attempt to account for the possible blocking actions of an adversary. Although this approach claims to account for the actions of an adversary, the actual outcome is not that different from the plans generated by programs like TACPLAN. The lack of difference between the two approaches is caused because the actions of an adversary are prescribed in advance which is similar to only

permitting a chess adversary to make forward moves. Infinitely many strategic choices can be made by an adversary that will not be covered by the prescribed action choices that are permitted by the tactical planning algorithm.

The POLITICS system (Carbonell, 1981) proceeds in the development plan by introducing goals that an adversary is trying to accomplish, such as blocking the current goal of the planning program. The adversarial goals used in POLITICS are statically defined and are based on preconceived notions of an adversary's intent. Basing the actions of an adversary on static predefined goals runs the risk of creating tactical blindness mentioned in the introduction to section 2.7. The actual goals of an adversary, much less how the adversary intends to accomplish those goals, may not be known during the contest or may be contrary to the expected goals (Crookall et al., 1987 and Narendra & Thathachar, 1989). POLITICS does provide a good example of a program in which a thinking adversary attempts to achieve specific goals.

TM (Erickson & Zytkow, 1988) is another goal based system for making tactical decisions. The program BOGEY is an external learning program which generates large numbers of simulations of tactical engagements with specific goals. BOGEY, like GINA in section 2.5, is another example of the utility of coaching programs that support existing domain specific programs through an external learning function.

The MARK I and MARK II systems (Davis, 1988a and 1988b and Davis et al., 1986) embody the mind set of an adversary in DEWT, Deterrence Escalation

control and War Termination, models. Currently the MARK systems have two decision models, East and West. While MARK I and MARK II attempt to capture differences in the mind sets between adversaries, the models are statically defined like the goals of the POLITICS program. These static mind sets are based on predefined conceptions of an adversary's intentions which may not be accurate and will not account for the modification over time to decision heuristics used by an adversary. One of the contributions that artificial intelligence can provide to military planning is reasoning that accounts for the temporal relevance of actions (Bonasso, 1988).

The use of multiple models to generate goals for different adversaries demonstrated by the MARK systems is a reasonable approach to separate the varying evaluation criteria used by various adversaries. An improvement to the MARK systems is accomplished by acquiring the evaluation criteria and traits of particular adversaries in specific situations along with the temporal relevance for each of the evaluation criteria of an adversary.

We mentioned in section 1.2 that geometric or visual relationships between domain resources are used by the inductive learning mechanism of the adversary modeling methodology. This means that military domains are required to represent the physical resources of the domain in a geometric manner. The services are trained to use standardized formations such as the column, echelon, line, and wedge formations (Department of the Army, 1986). Typical board wargames make use of a hexagonal map for plotting the movement of resources. By using such map

representations, the use of specific formations by a particular commander can be acquired by our adversary modeling methodology.

2.8 Political and Economic Domains

Top level policy decisions made by government officials are also affected by prior knowledge about an adversary's goals and strategies (Cimbala, 1987). Government negotiators attempt to discover the limitations faced by an adversary to accomplish the optimal outcome of an agreement or treaty between the two adversaries. The use of knowledge of the political mind set for a geographic region permits negotiators to interpret information from the proper perspective (Cimbala, 1987).

The political use of adversary modeling can also be extended to corporate politics. We can acquire patterns from adversarial actions such as corporate takeovers or mergers. This knowledge enables corporate entities to be better defended or prepared for such adversarial actions.

Other domains that are concerned with the geometric relationships of elements in the domain can benefit directly from the geometric pattern acquisition phase of the adversary modeling methodology. Examples of such domains include city planning and architecture. For the city planning domain, patterns of streets from other cities which have desirable and equivalent traffic patterns can be acquired for use in the current city. The primary benefits to these domains will be economic savings and time savings for the reduced complexity of the design phase for individual projects.

2.9 Religion Domain

The importance of knowing the intentions and strategic style of an adversary has been recorded throughout history. An early example of these writings comes from the first century A.D. when the apostle Paul (1 Corinthians 9:19-22) writes:

> that I might win the more. And to the Jews I became as
> a Jew, that I might win Jews; to those who are under the
> Law, as under the Law . . . to those without law, as
> without law . . . To the weak I became weak, that I
> might win the weak.

Paul is stating that he will model the lifestyle of the people he is attempting to win. The training centers for the military at the National Training Center, Red Flag, and the Top Gun schools are modern examples of the approach used by Paul.

2.10 Chess Domain--Our Application Area

We mentioned in section 2.5 that games like chess can be thought of as abstractions of actual military contests. Although the outcome of these games may be viewed as trivial compared to the outcomes of military conflicts, personal reputation, and financial prosperity are the stakes of such games for expert game players.

In the remainder of this section, we demonstrate the use of adversary modeling by humans in the chess domain and the philosophical and psychological foundations that enable the adversary modeling methodology to operate in the chess domain. Finally, we present a detailed background of past and current computer chess programs.

2.10.1 Human Use of Adversary Modeling in Chess

Before playing for the World Chess Championship against Capablanca who was considered to be invincible and was appropriately called "The Chess Machine", Alekhine studied and analyzed Capablanca's prior games (Schonberg, 1973). From his study, Alekhine was able to determine a weakness in Capablanca's play that he was able to exploit to become the new World Chess Champion. Other chess masters including the World Chess Champions Botvinnik, Tal, and Kasparov prepare for chess matches by rigorously studying the previous games of their opponents (Horowitz, 1973).

Nunn (Nunn & Griffiths, 1987) describes a game against Øst-Hansen in which both players based their game strategy on expectations of their adversary's playing style. These expectations were formed from prior experience against the adversary. Although no one was able to capitalize upon the information, it has been noted that prior to his World Championship match against Spassky, Fischer varied his standard opening for white from e4, pawn to King four, only three times during tournament level play (Evans, 1970).

Several of the chess grandmasters, including Lasker and Tal (Horowitz, 1973 and Schonberg, 1973), have been described as psychological players. These players would intentionally make the move that would be most disturbing to their opponents, even though such moves were frequently not the best tactical choice. Levy (1984) has described his style of play against computer chess programs similarly, stating that his move choices are often made to confuse the computer.

The advent of intelligent programs in various game domains, including chess, has caused humans expert game players to consider the computer as an adversary. Subsequently, human competitors who will face an intelligent computer program as an adversary have created models of adversary strategy and playing style that represent the computer game programs. Certain human competitors are considered to be specialists against computer styles of play.

A famous example of this specialization is the series of bets which David Levy (1984) has had with the computer chess community. In these bets, Levy claims that no computer chess program will be able to beat him. The first bet which had a ten year duration was won by Levy, however, DEEP THOUGHT has defeated Levy in the latest of the bets. A recent discussion on the rec.games.chess news network by Feng-Hsiung Hsu indicates an increase in the number of human opponents of computer chess programs using "anti-computer" playing styles against programs entered in major chess tournaments.

When Mike Valvo defeated DEEP THOUGHT, the reigning computer chess program champion for the past three years, in a two game postal chess match via electronic mail, it was noted (Newborn & Kopec, 1989) that Valvo had observed DEEP THOUGHT play a number of games and had acquired the playing style of DEEP THOUGHT. The chess program did not have a similar opportunity. Berliner (1988) claims that various chess players have asked for the previous game records of HITECH, the 1989 computer chess program co-champion, so that they might be prepared for a possible match.

2.10.2 Importance of Pawn Structures and Chess Openings

We pay special attention to pawn formations during our chunk learning process. Philidor has stated that the win or loss of a game depends on the quality of the pawn positions during the game (Pachman, 1975). The strength and weakness of pawn formations is of paramount importance (Lasker, 1973). Pawn formations can be used to reveal information concerning positional advantages (Reinfeld, 1987). Furthermore, three of the six criteria for determining the character of a position given by Capablanca relate to the effectiveness of the pawns in the game (Pachman, 1975).

Pawn formations contain information about the positional nature of the chess game (Soltis, 1976). The geometric patterns of pawns displayed by a specific adversary can be used to identify his playing style. By definition we can recognize open and closed board positions solely from the absence or presence of long pawn structures. The general playing style of the adversary is indicated by the predominant style displayed by the adversary in the studied games. Adversaries will tend to remain with the particular style that they feel most comfortable in playing. Chigorin played only open style games (Pachman, 1975). Our own analysis of the games of Botvinnik and Spassky has indicated that each of these chess grandmasters prefers a particular style, closed and open respectively.

Openings also play an important role in determining the positional advantage of the game. Current computer chess programs utilize a book or database of opening moves which have a duration from two moves to more than ten moves. The evaluation algorithms which enable chess programs to estimate the strength of the

current game position are not used until the book is exited. Typically, exits from opening books occur only when an adversary makes a move that is not contained in the book or the current opening line of play is exhausted. This means that current computer chess programs can find themselves from ten to twenty percent of the way through a game, for a game lasting fifty moves, before they start to calculate the value of the current position.

The opening sequences of play that are used as the opening book for current chess programs are stored verbatim. Mednis (1990) has demonstrated that particular opening formations can be achieved through a wide variety of the actual move order. A statistical analysis of the opening move sequences displayed by an adversary accounts for opening move order variations by finding the upper bound, lower bound, and mean time of specific moves.

Selecting an opening line of play is usually performed at random or in response to the adversary's choice of opening move. By studying the playing style and previous games of an adversary, specific opening lines that are well known to an adversary are identified. This knowledge can be used by a chess program to emulate human chess experts by choosing openings that are not as well known to the adversary. Nunn (Nunn & Griffiths, 1987) avoids the Pelikan variation of the Sicilian Defense in one of his games against Anthony because Nunn knows from experience that Anthony is a Pelikan specialist. Other examples of selecting opening lines of play based on the prior performance of an adversary are given by Nunn.

Acquisition of the opening sequence knowledge held by the adversary will enable current chess computer programs to select lines of play which are strategically

advantageous. Holding (1985) supports our perspective by stating that we should use information about players playing style to disrupt their normal play.

2.10.3 Philosophy

Dennett (1987) states that current chess programs concentrate their time on branches of a game tree which represent the adversary's best response as viewed by the program. While this would appear to be a reasonable approach, Dennett elaborates by stating that a rational adversary might have a higher ulterior motive than avoiding defeat in chess. For example, a young Soviet chess player with a rating of master is performing well in a European chess tournament. The young Soviet has acquired a score of 8.5 points and is about to face the tournament champion, a fellow Soviet grandmaster who has scored 11 points thus far. The grandmaster and young master both believe that the grandmaster will win the final game. However, no other player in the tournament has a score greater than 9 points. The Soviet grandmaster realizes that an outcome of a draw with himself will greatly raise the young Soviet master's chess rating without damaging his own rating and might improve the young master's confidence and future playing ability. Therefore, the Soviet grandmaster who has already won the tournament by virtue of his previous victories offers the younger master-level player a draw midway through the game.

Another example of the problems faced by computer programs in dealing with the intentionality of human adversaries is provided by Levy (1984) who claims that he will intentionally choose inferior moves when playing against a computer. The inferior move choices are selected by Levy because he believes that computer chess

programs have greater difficulty in playing a winning game against such an irrational style of play.

The Levy example above demonstrates one of the traits of human adversaries. Humans, especially when competing against a machine, attempt to mislead their adversary about their intentions and capabilities. We must be able to distinguish between relevant information and misinformation about an adversary's playing style. Incorporating the detection of misinformation from an adversary into the inductive learning mechanism adds an additional heuristic element. The addition of another heuristic element reduces the probability of acquiring a valid adversary model.

The effect of an adversary's misinformation attempts is negated by selecting previous domain performances that limit the possibility of misinformation. By only using the previous games of an adversary from tournament play in which the adversary has something at stake, misinformation is eliminated. For our application in the chess domain we use the World Championship Match games of adversaries taken from the compendium by Gelo (1988).

2.10.4 Psychology

Chase and Simon (1973) have extended the research performed by de Groot which claims that the depth of game tree search and the total number of moves considered is approximately the same for both novices and chess masters. However, the ability to recall chess positions after a five second presentation of the positions is markedly different. Chess masters are able to reconstruct chess positions involving twenty or more pieces with complete accuracy. Chess novices are only able to place

four or five of the pieces correctly. The difference between the masters and the novices disappears when the subjects are asked to reproduce random patterns of chess pieces. Each group can only place three or four of the pieces from the random board configurations. The apparent decrease in ability by the master level players is attributed to the fact that the random configurations of pieces contained no inherent domain knowledge.

Based on Miller's (1956) hypothesis, the chess masters are forming higher level chunks composed of groups of pieces from the game positions. These chunks contain domain specific knowledge from the prior experience of the chess masters. Chase and Simon (1988) report that human chess masters store from 10,000 to 100,000 chess patterns.

The skill of chess masters increases as more patterns are acquired. Novice chess players move away from slow deductive reasoning towards the fast perceptual processing used by masters as more and more patterns are learned (Chase & Simon, 1973). The chunking theory of learning (Laird et al., 1986) supports this viewpoint in stating that performance improves via the acquisition of knowledge about patterns and chunks. Richard Seltzer, the father of fourteen year old chess master Bobby Seltzer, in a personal communication relates that Bobby's increase in skill is reflected by an improved perception of the board position and subsequent position evaluation.

Bratko et al. (1986) state that natural positional moves are closely related to the chunk structures of a game position. Specific chunks cause chess players to generate tactical and strategic plans. The chunk displayed in Figure 4 will cause an

adversary to immediately consider plans for a back-row mate with either the queen or one of the rooks. Back-row mate plans result from the knowledge contained in the chunk concerning the opponent's king's limited mobility.
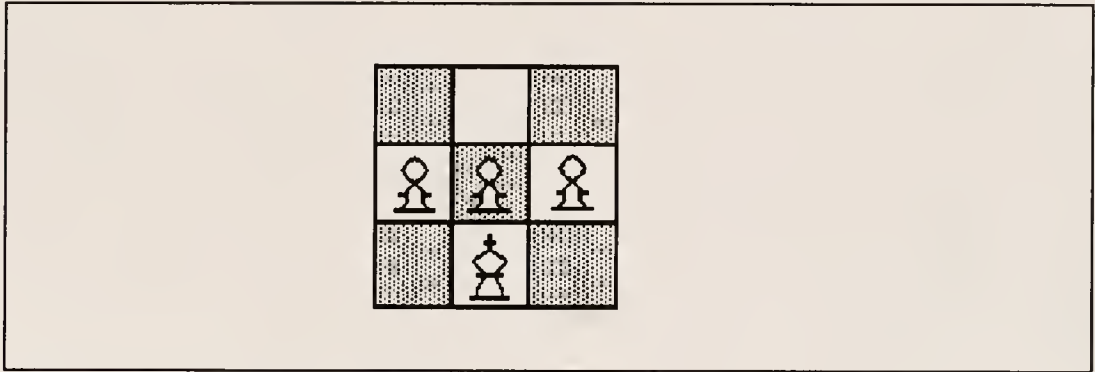
Figure 4: Sample cognitive chunk.

Humans recall chess chunks through familiarity with common board configurations. Simon and Gilmartin (1973) developed MAPP which reconstructs chess positions by simulating human eye movements recorded in their study. Simon and Gilmartin's research shows that humans tend to group or chunk together pieces which share geometric proximity. We will use geometric proximity as the primary filter for selecting chess pieces to be formed into chunks. Proximity is one of the Gestalt principles of organization (Anderson, 1980) along with similarity, good continuation, and symmetry. We will also utilize the Gestalt principles of similarity and good continuation to construct chunks. Chase and Simon (1973) found that proximity and piece color, or similarity, were the two strongest factors affecting chunk pattern recall by their chess master subjects.

The significant use of patterns of game pieces by chess masters has led us to make the following three part hypothesis:

- Chess masters acquire several thousand patterns of pieces that are used to evaluate a game position.

- Chess masters will continue to use a specific strategy while winning games and tournaments.

- Chess masters will tend to reduce the complexity of a game position by moving board positions that do not contain familiar patterns into positions that do contain familiar patterns.

We present the general hypothesis for all adversarial domains in Chapter 3. This version of the hypothesis is the chess domain translation from the general hypothesis and is presented here to examine the underlying foundations. The first part of our hypothesis is a direct result from the research of Chase and Simon examined above.

The second part of our hypothesis is supported by the Reinforcement Theory of psychology and is an extension of the results from various psychological experiments (e.g., the water jug problem (Anderson, 1980) and the 1023-choice task (Laird et al., 1986)) which indicate that human subjects will continue to use a learned strategy for solving a class of problems even when a more efficient strategy would also solve a specific problem.

While the third part of our hypothesis is intuitively appealing, direct support comes from the research of Horgan et al. (1989) and Saariluoma (1984) which demonstrates the use of chunking and analogy to similar positions by chess masters in solving chess problems. The chunking of several pieces into a known pattern produces a cognitive economy while evaluating complex board positions.

The search complexity of game trees is reduced as a result of the application of our hypothesis. The strategic and tactical decisions of an adversary can be accurately predicted because the adversary will tend to recreate familiar patterns to reduce the cognitive complexity of evaluating the current game position. A practical example of the use of familiar patterns is demonstrated in the 1978 Budapest tournament game between Pinter and Bronstein (Roycroft, 1988). This game resulted in a KNKBB endgame. Chess literature had previously claimed that a position known as the Kling and Horowitz position (Kb6Nb7) will result in a draw for the KNKBB endgame. Pinter's knowledge of the Kling and Horowitz position led him to consistently maneuver to create this position. Pinter succeeded in forming the Kling and Horowitz position three different times in three different corners of the board. However, Pinter's haste to create the chunk caused him to make inferior moves nearly thirty percent of the time according to the 250,000,000 position BBN database (Roycroft, 1988) which exhaustively solves the KNKBB endgame in an optimal number of moves.

Computer chess programs need to simulate the cognitive economy of chess masters to continue to improve in performance. Lenat et al. (1979) state that intelligent systems must perform expectation filtering, using predictions to filter unsurprising data. Such a filtering process is provided in adversarial domains by the ability to predict adversary moves in specific situations.

2.10.5 Computer Chess and Artificial Intelligence

The first chess "computer" was the automaton constructed by Torres y Quevedo circa 1890 (Bell, 1978). Torres y Quevedo's chess machine was capable of

accurately playing certain variations of the KRK, King and Rook versus King, endgame.

Since Claude Shannon's (1950) paper on the requirements for a chess playing program, many artificial intelligence researchers have devoted themselves to the creation of master level game playing programs. Shannon defines two main strategies for developing chess playing programs. These strategies are the type A strategy which generates all possible moves to a certain depth of the game tree and then uses an evaluation function to choose the best move and the type B strategy which uses heuristics for suggesting moves without an exhaustive search.

The type A strategy chess programs dominate the type B strategy competitors. Northwestern University's type A chess program CHESS4.X (Frey, 1983), were X is a version number, reigned during the 1970s as the North American computer chess program champion and was the World Computer Chess program champion from 1977 until 1980 (Kopec & Newborn, 1987). The three type A programs BELLE, HITECH, and DEEP THOUGHT have each set the high water mark for chess program playing strength at USCF ratings of 2200, 2400, and 2500 respectively. These three programs have also been the North American computer chess program champions from 1985 to the present (Newborn & Kopec, 1989).

The chart in Figure 5 shows the progress in playing strength of chess programs with respect to search speeds. The bottom axis is logarithmic which means that the linear progression in strength of chess programs during the 1970s was accomplished by an exponential increase in search speeds. Recent progress depicted by the dashed
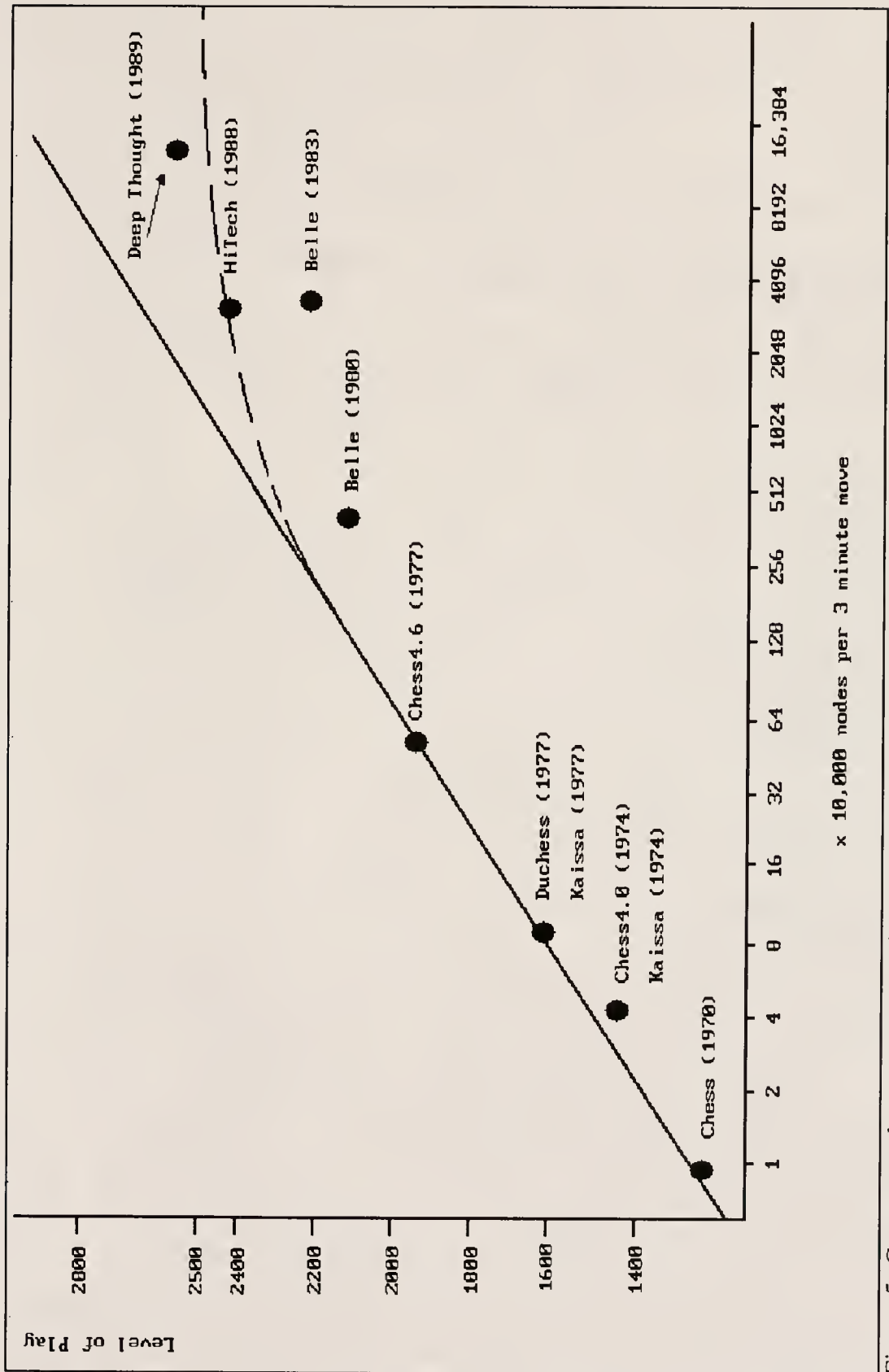
Figure 5: Computer chess program ratings.

curve reflects the hyper-exponential increase in search speeds required to maintain the linear growth in playing ability. BELLE, HITECH, and DEEP THOUGHT each use specialized hybrid architectures to achieve the dramatic increases in search speed depicted in Figure 5 (Ebeling, 1986). HITECH uses a VLSI architecture with fine grained parallelism to perform pattern matching which is cognitively similar to the pattern recognition performed by human chess masters. DEEP THOUGHT (Newborn & Kopec, 1989) uses special VLSI circuitry to achieve a search rate of 720,000 position nodes per second.

Although type A strategy chess programs are the best performing chess programs, their performance against grandmaster rated human players has been abysmal. In an exhibition round robin tournament played in October 1989 at Boston, four human grandmasters crushed four of the top computer chess programs, including HITECH and DEEP THOUGHT (Hamilton, 1990), with a score of 14.5 to 1.5. World Chess Champion Gary Kasparov defeated DEEP THOUGHT in a two game match played in New York prior to the Boston tournament. Kasparov commented that DEEP THOUGHT lacked experience and was unable to analyze the reasons for some of the moves he made (Geitner, 1989).

Berliner and Goetsch (1984) state that the performance of chess programs is proportional to the amount of knowledge used by the program's evaluation function. Bratko and Michie (1980) demonstrated the need to use domain specific knowledge to solve certain domain tasks by having CHESS4.5 and the AL1 program try to solve the KRKN endgame. CHESS4.5 was unable to solve two of the three chess

problems used in Bratko and Michie's research. However, AL1 which uses a type B heuristic method with domain knowledge was able to solve all of the endgame problems.

Type B strategies vary in their approaches to playing chess. The first heuristic method uses plans or goals that have been constructed from domain specific expertise to solve chess problems. AL1 is a goal based chess program which solves KRKN endgames. LEBL (Tadepalli, 1989) uses goals during King-Pawn endgames to reduce search complexity. Wilkins (1980 and 1982) also uses domain specific knowledge to produce plans which his program PARADISE tries to achieve. Goal based planning allows PARADISE to accurately play quiescent middle game positions. The use of goals to achieve higher playing performance in chess programs simulates the cognitive method of means-ends analysis used by humans in certain problem solving tasks (Anderson, 1980).

Another heuristic approach which is similar to the cognitive methods analyzed by Chase and Simon, described in section 2.10.4, simulates the human cognitive process of chunking. This heuristic approach tries to identify common configurations or chunks and then uses these chunks to identify analogous solutions to chess problems. Quinlan's ID3 program (in Michalski et al., 1983) tries to inductively classify equivalent board configurations in the KRKN endgame. The ID3 induced patterns are used to determine if the King-Rook side has a winning position within a specified number of moves. Currently the program has solved the KRKN endgame for all positions which can be won in three ply. Campbell's (1988) CHUNKER also

takes advantage of board configurations to produce chunks which are used to solve King-Pawn endgames.

Each of the heuristic methods described above have been implemented on endgame or middle game portions of chess problems and deal primarily with tactical solutions to chess problems. The knowledge available to chess programs can be significantly augmented by extending the research of Quinlan and Campbell in the following ways:

- Collect chunks from complete chess games instead of just the endgame segment.

- Use the chunks of pieces to predict an adversary's tactical and strategic movement decisions.

2.10.6 Justification for Chess Domain Application

Shapiro (1987) presents several reasons for choosing chess as an experimental test-bench.

1. The game constitutes a fully defined and well-formalized domain.

2. The game is sufficiently complex to challenge the highest levels of human intellectual capacity.

3. A large body of knowledge has been accumulated and is available in literature form.

4. A generally accepted numerical scale of performance is available so that increases in performance can be rated.

Our application domain needs to be sufficiently complex so that exhaustive search

is not a feasible solution. Typical chess game trees consist of 10^{43} nodes (Berliner & Goetsch, 1984) which prohibits exhaustive search solutions given current technological capabilities. Because chess is a well understood domain and there is an accepted numerical rating scale, performance improvements contributed by the adversary modeling methodology can be evaluated.

CHAPTER 3
ADVERSARY MODELING METHODOLOGY

In this chapter, we examine in detail each of the functions required to create an adversary model from the adversary modeling methodology. The foundation for our adversary modeling methodology is the three part hypothesis:

- Relevant patterns and plans are learned by the adversary from practical experience in the domain.

- The strategic and tactical actions of an adversary are repeated as long as the actions produce a positive result.

- An adversary tends to reduce the complexity of a situation to enable a more precise strategic evaluation of the situation by using the plans and patterns already acquired in the domain.

Evidence supporting this hypothesis and the specific chess domain translation are presented in section 2.10.4.

To accurately predict the strategic and tactical plans of an adversary, we must first be able to perceive and evaluate the domain the same as our adversary. This means that we need to acquire the evaluation criteria used by our adversary. The first and third parts of our foundational hypothesis indicate that all competitors in adversarial domains make use of patterns that have been previously acquired in the domain for their evaluation process.

45

3.1 Knowledge Acquisition of Chunks

For chess, the patterns involve both the geometric patterns of pieces and the textual patterns related to the actual moves that are executed. The geometric patterns correspond to chunks that are stored in visual memory, while the textual patterns correspond to chunks that are stored in the adversary's verbal memory.

3.1.1 Defining the Chunks to be Acquired

We rely on the Gestalt principles of organization and the research of Chase and Simon described in section 2.10.4 to construct our definition of the elements that are to be contained in a geometric chunk. Because we are trying to find chunks of pieces that are used to reduce the complexity of a situation, our chunks necessarily have a minimum of two pieces. Although chunks need only two pieces to satisfy our definition, chunks are generally required to contain a minimum of three pieces to be used in our predictions of probable adversary moves. Chase and Simon (1973) indicate that the chunks used by chess masters contain three to four pieces. Two piece chunks are strategically significant in the opening game segment of chess due to the minimal number of pieces that have moved from their starting positions and in the end game segment due to the previous elimination of pieces. However, two piece chunks in middle game positions often occur without effect from the strategy or cognitive simplification efforts of the adversary. The chunking mechanism emphasizes chunks which contain a minimum of three pieces.

Based on the principal of proximity, chunks contain only pieces which are adjacent to another piece. Pieces contained in a chunk are all of the same color to

satisfy the principle of similarity. Requiring chunks to be of a single color, namely the pieces played by the adversary, restricts the contents of the chunk to be under direct control of the adversary. An adversary cannot control the move choices of an opponent except for inter-locked pawn chains and therefore cannot use a chunk with mixed colors of pieces for long range strategic planning.

Research performed by Church and Church (Holding, 1985) demonstrates that human chess players have difficulty in processing information from pieces on diagonals. From this research we only chunk pieces by proximity which are adjacent along either the horizontal axis or vertical axis of the playing board. Other adversarial game domains like checkers can rotate the major axes to include the diagonals and exclude the horizontal and vertical axes.

The Gestalt principle of good continuation or meaningfulness is used to chunk pieces which are capable of affecting the strategic meaning of a chunk from the diagonals. The bishops, queens, and king are all included in a chunk if they are diagonally adjacent to another piece in the chunk. Meaningfulness is also used to include pawns which are diagonally behind, to reflect the movement capabilities of the pawn piece, a piece in a chunk.

Furthermore, we need to define the maximum size of each chunk to build an efficient knowledge representation structure for storing the chunks. Humans compose complex chunks hierarchically from smaller, less complex chunks. We use a maximum chunk size of a four-by-four, or sixteen square, board area. All of the chunks collected by IAM from a twelve game analysis of M. Botvinnik are shown in

Figure 6. From these chunks we can see that a sixteen square area is sufficient to hold all of the chunks found. We increased the maximum size of defined chunks to five-by-five and six-by-six to measure the affect of a larger chunk size. Our research indicated that only one additional chunk was captured during an analysis of twenty games played by Botvinnik (see Section 4.2.4, Figure 22). This marginal increase in acquired chunks does not merit the increased cost in space requirements for larger chunk definitions. The sixteen square size chunk is the minimal implementation that also permits the use of a chunk to acquire additional knowledge we deemed useful, including the pieces remaining at the end of each game.

Because of the significance of pawn formations in the analysis of chess positions, we acquire a second type of geometric chunk which is aimed at acquiring pawn formations. As noted above, pawn formations are stable and an adversary can rely on the presence of an opponent's pawns in a chunk. Our similarity principle is modified to collect all similar pieces, i.e. pawns, instead of similar colors. The proximity and meaningfulness principles remain the same for pawn chunks. Because pawn structures can extend over a large area of the game board, we increase the size of the maximum defined chunk from a sixteen square area to a thirty-six square area. Support for increasing the maximum chunk size for the pawn formation chunks was found when several pawn structures were acquired during our research which would have exceeded the sixteen square area used in the standard geometric chunk definition.
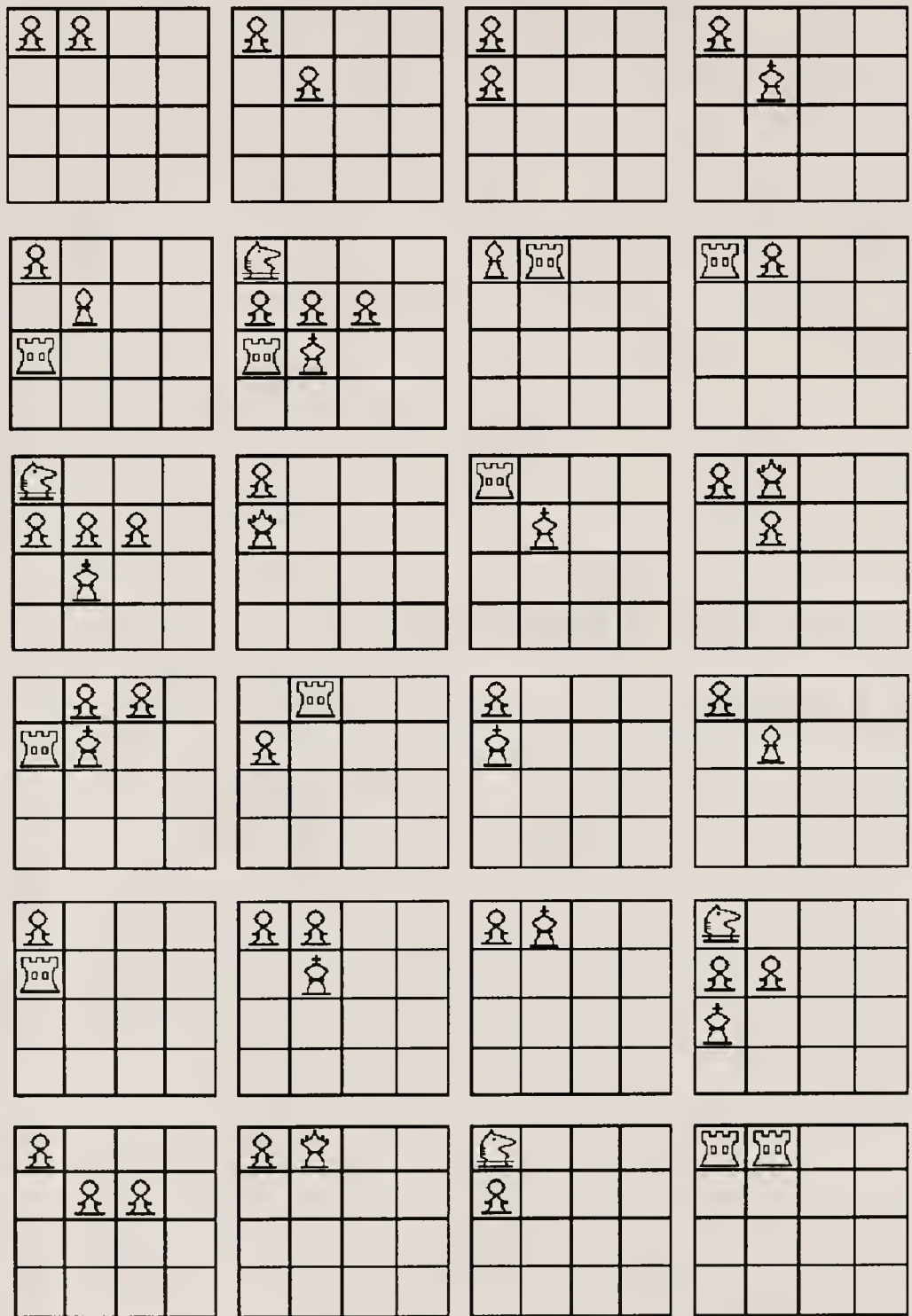
Figure 6: Chunks acquired for Botvinnik.
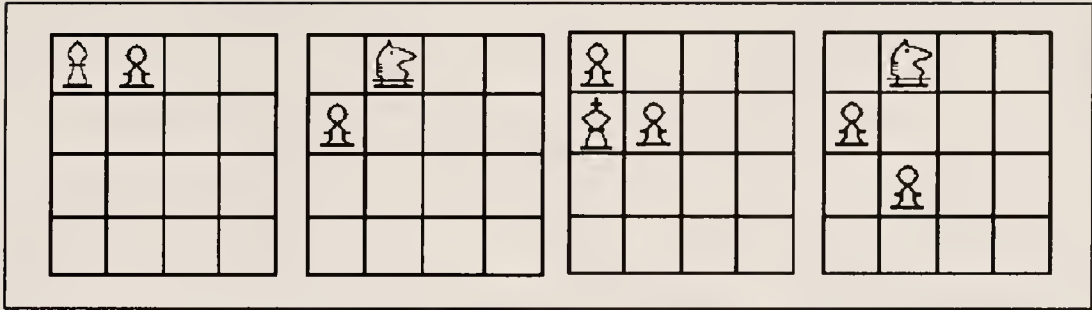
Figure 6--continued.

We have emulated the cognitive perceptual processes of human chess players to define the composition of geometric chunks. Our definition of the elements which are viable chunk members acquires chunks which are easily perceived and which contain the maximum amount of knowledge concerning the current game situation.

Our definition of textual chunks is less complicated. Human chess players remember verbatim dozens of opening lines of play that they have experienced. We emulate this rote learning by chunking the first five moves made by the adversary verbatim from the game records of the studied games. Our research indicates that chess masters tend to leave an opening line of play between the fifth and tenth move of the game. We are interested in acquiring knowledge that is high in quality and reliability and therefore we rote learn only the first five moves. One of our experiments considered the effect of increasing the textual chunk size to ten and fifteen moves and the results of this experiment are detailed in Chapter 4.

3.1.2 Acquiring Geometric Chunks

From our background study presented in Chapter 2, we note the significance of perceptual processing in the development of chess expertise. We take advantage of this aspect of expert chess performance by viewing the game board and pieces as

an image to be processed into chunks. By viewing the game board as an image, we consider via analogy previously defined computer vision techniques for processing images and collecting information.

The convolution operator is used in the image algebra (Ritter et al., 1988) of computer vision to collect information from the area surrounding a specific pixel, which is analogous the proximity principle we use for defining chunks. For games, a homomorphism can be drawn between pixels in an image and pieces on a game board. The meaningfulness principle that we use in our chunk definition requires us to capture knowledge from different areas surrounding a piece, depending on the current piece being analyzed. The technique of varying templates enables us to alter the summing template used by the convolution operator to correspond to an individual piece. This permits us to collect information about the proximal pieces that is dependent upon the central piece.

We maintain an internal representation of the game board in an eight-by-eight array with subscripts 'i' and 'j'. The first subscript denotes the column and the second subscript denotes the row. Our internal representation is similar to the one described by Shannon (1950) with negative values assigned to non-adversary pieces and the numbers one through six representing the pawns, knights, bishops, rooks, queens, and kings respectively. Unoccupied squares are denoted by the value zero. A board position and its corresponding internal representation are demonstrated in Figure 7. The piece value at each board location is denoted by the algebraic symbol $X_{i,j}$.
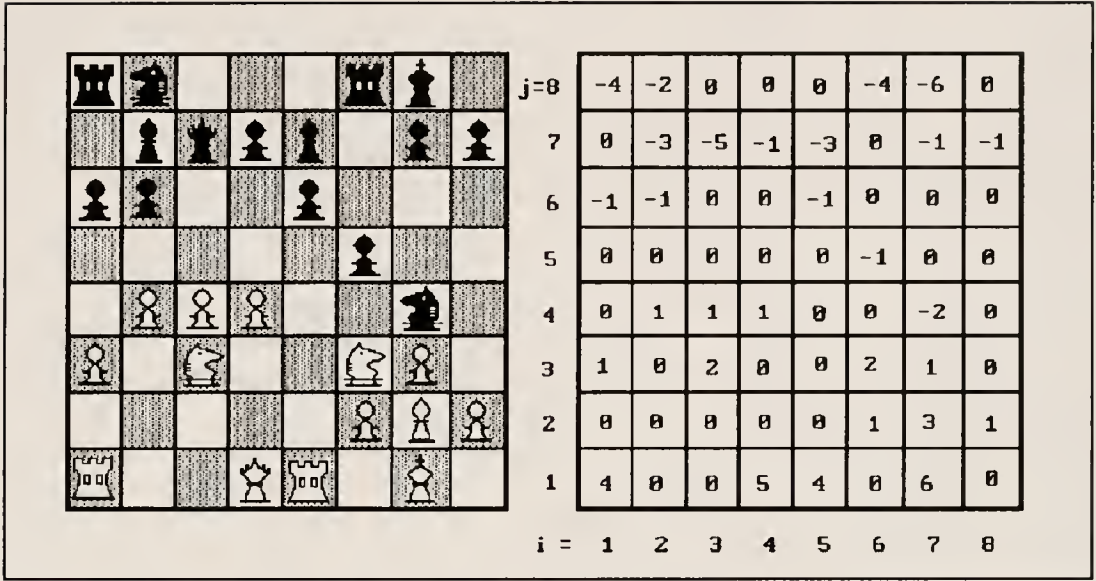
Figure 7: Internal representation of a board position.

The three different templates used in our implementation of the convolution algorithm are shown in Figure 8. Each template can be thought of as a three-by-three array with the center location denoted by the subscript (0,0). Other template subscripts range from negative one to one. The general template in Figure 8 is used for bishops, queens, and the king with the other pieces being convolved by the appropriately named template. The pseudo-code in Figure 9 demonstrates how the convolution operation is performed. When an adversary's piece is detected, the appropriate template is loaded into a copy of the current template--TP. The values depicted in the templates were a proximal piece is located are summed and stored in the return array Y using an inclusive or operator--⊕. We have defined the predicate function, ON_BOARD, to verify that the array subscripts 'i + M' and 'j + N' correspond to a valid board location. The return array Y is an eight-by-eight copy of the internal board representation with all values initially set to zero. This

4	8	16
2	╱	32
1	128	64

General

4	8	16
2	╱	32
0	128	0

Pawn

0	8	0
2	╱	32
0	128	0

Rook, Knight

Figure 8: Convolution templates.

```
Case X_{i,j} of
    3, 5, 6      : TP ← the general template      ;; (See Figure 8)
         1       : TP ← the pawn template
      2, 4       : TP ← the rook-knight template
    otherwise Return        ;; Non-adversary piece, no need to convolve
End Case
For N = -1 to 1 do
    For M = -1 to 1 do
        If ON_BOARD(i+N, j+M) and X_{(i+N),(j+M)} <> 0
        Then Begin
                Y_{i,j} ← Y_{i,j} ⊕ TP_{N,M}
                Y_{(i+N),(j+M)} ← Y_{(i+N),(j+M)} ⊕ TP_{-N,-M}
            End Begin
```

Figure 9: Pseudo-code to perform convolution.

effectively captures all pieces satisfying our chunk definition qualities and returns their relational position to the center piece location in Y.

The internal representation of the game board is updated after each move that is read from the game record. For example, if the move 'a1-c1' is read to move the rook, in the lower-left corner of the board in Figure 7, two squares to the right,

then the value at $X_{3,1}$ would be set to four and the value of $X_{1,1}$ would be changed from four to zero. Currently we use a modified algebraic notation for recognizing the moves played during the game. Other notations can be easily translated to algebraic notation for use by the IAM program. The chunks we desire to capture are the ones that have just been formed by the adversary, which improves our capability for predicting definite adversary actions. Therefore, we collect chunks following the adversary's move.

The appropriate step function shown in Figure 10 is applied to the board representation X and is stored in an eight-by-eight array Y. Pseudo-code which reduces the complexity of the ensuing board search by removing all pieces that do not satisfy the appropriate similarity constraint is shown in Figure 11. We use the step functions to increase the speed of the search algorithm which locates pieces that can be chunked together. The convolution operation shown in Figure 9 is then performed at the site of the current board search. If any neighbors are identified as belonging to the chunk, then the convolution operator is applied iteratively to all related neighbors. This acquires the chunk as soon as the left-most and bottom-most piece, with respect to the a1 square, is found.

Chunks of pieces which have never been moved by the adversary do not represent any significant new geometric relational or strategic knowledge. To reduce the total number of chunks being stored, we discard chunks which have greater than half of their pieces still in their original starting positions. To further reduce the storage requirements of the adversary modeling methodology we take advantage of
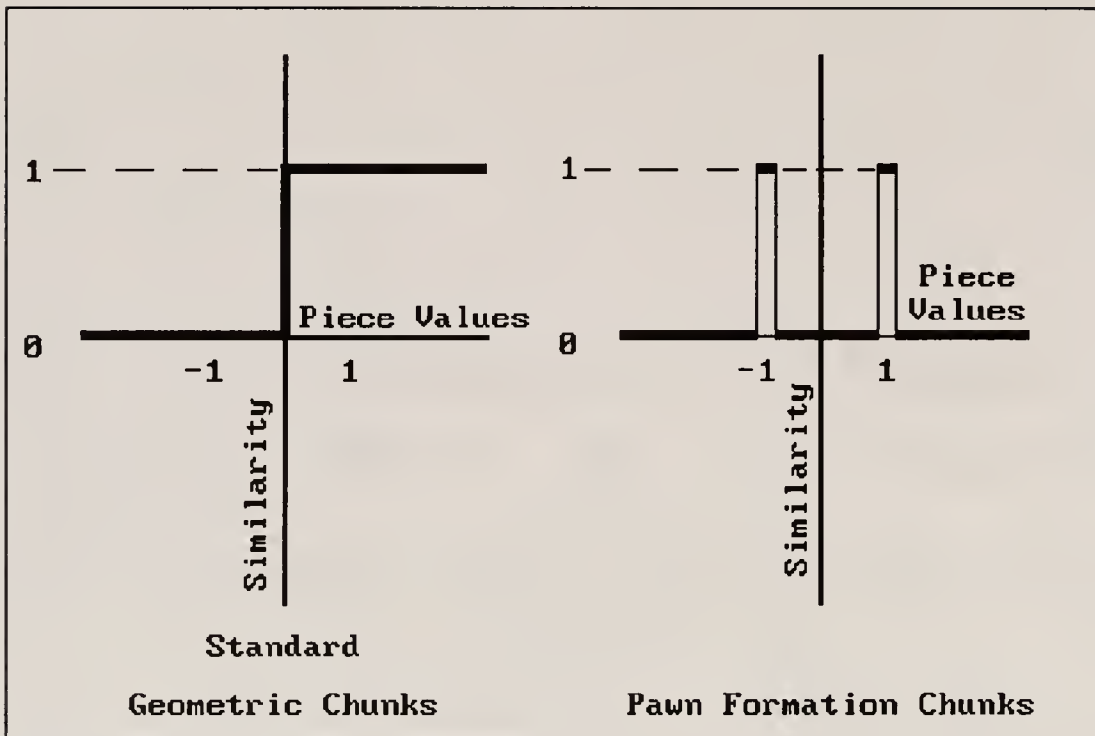
Figure 10: Step functions.


```
For I = 1 to 8 do
    For J = 1 to 8 do
        If gathering standard geometric chunks AND X_{I,J} > 0
        Then Y_{I,J} ← 1
        Else If gathering pawn formation chunks AND ABS(X_{I,J}) = 1
            Then Y_{I,J} ← 1
            Else   Y_{I,J} ← 0
For I = 1 to 8 do
    For J = 1 to 8 do
        If Y_{I,J} = 1 Then perform convolution
```

Figure 11: Pseudo-code to perform step function.

chunk durations. If a chunk is created on the tenth move of a game and persists until the twentieth move, then only one chunk is created instead of ten chunks. The

times of a chunk's inception and subsequent dissolution are stored with each chunk. We also use the time tags associated with each chunk to eliminate storing a chunk which is dissolved by the adversary and then recreated at a later time during the same game. Pseudo-code to perform the chunk comparisons, which produce the storage savings just mentioned, is shown in Figure 12. The variable 'C' represents the double linked list of chunk records with the subscript 'I' denoting the position in the list and the variable 'D' represents the most recent chunk that has been identified for the current game.

```
For I = 1 to the total number of chunks created for this game do
    If C_I = D
    Then Begin
            If D.start_time < C_I.start_time
            Then C_I.start_time ← D.start_time
            If D.end_time  > C_I.end_time
            Then C_I.end_time ← D.end_time
            D ← 0
            Exit the loop
    End Begin
```

Figure 12: Pseudo-code to perform chunk coalescing.

An example of the chunks identified by the convolution operator and the convolution values returned is shown in Figure 13 where the chunks have been acquired for the white player. The position shown in Figure 13 is taken from the fifth game of the 1951 World Championship match between Botvinnik and Bronstein after white's eighteenth move. The three chunks acquired by the convolution
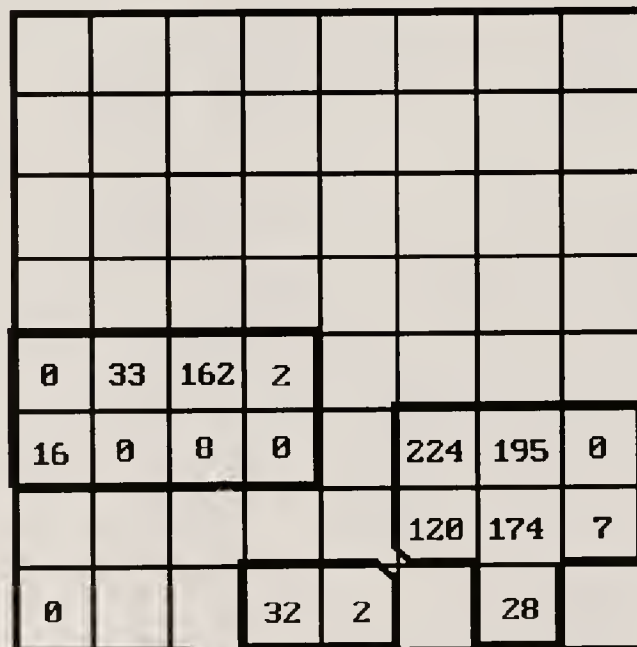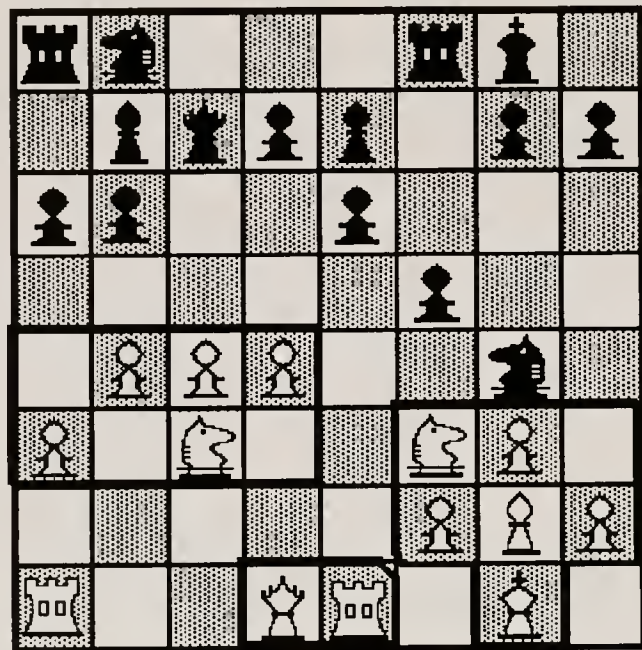
Figure 13: Chunks on a chess board and their convolution values.

algorithm are outlined in the figure. The pawn structure in the middle-left of the board could become a strong factor in the endgame and possibly create a passed pawn. This knowledge is available from the geometric relationship of the pieces in the chunk. The other two chunks are a standard offensive configuration (i.e., the queen and rook) and a standard defensive configuration (i.e., the fianchettoed bishop and castled king) that are found in many chess games.

The generality of the geometric chunk acquisition technique is demonstrated by applying the convolution operator to the adversarial domain of the Go game. Because only one type of piece is used, the general template from Figure 8 is the only template used in performing the convolution. The results of the convolution algorithm applied to a section of the Go game board are shown in Figure 14. This chunk represents the Go formation called eyes. Eyes are an important defensive strategic feature in Go. Other Go formations such as ladders, knots, false eyes, and tiger's mouths are also acquired by the adversary modeling methodology's convolution algorithm.

3.1.3 Acquiring Textual Chunks

The background research presented in Chapter 2 indicates that chess masters make use of their verbal memory to recall specific opening sequences of play. Our adversary modeling methodology acquires this facet of expert chess performance in addition to the geometric piece chunks.

Chess masters can recall verbatim particular opening sequences of play, which indicates the use of rote learning. We emulate the cognitive process of rote learning

The heavy black line represents the edge of the game board.

Values in squares correspond to pieces located at game board intersections
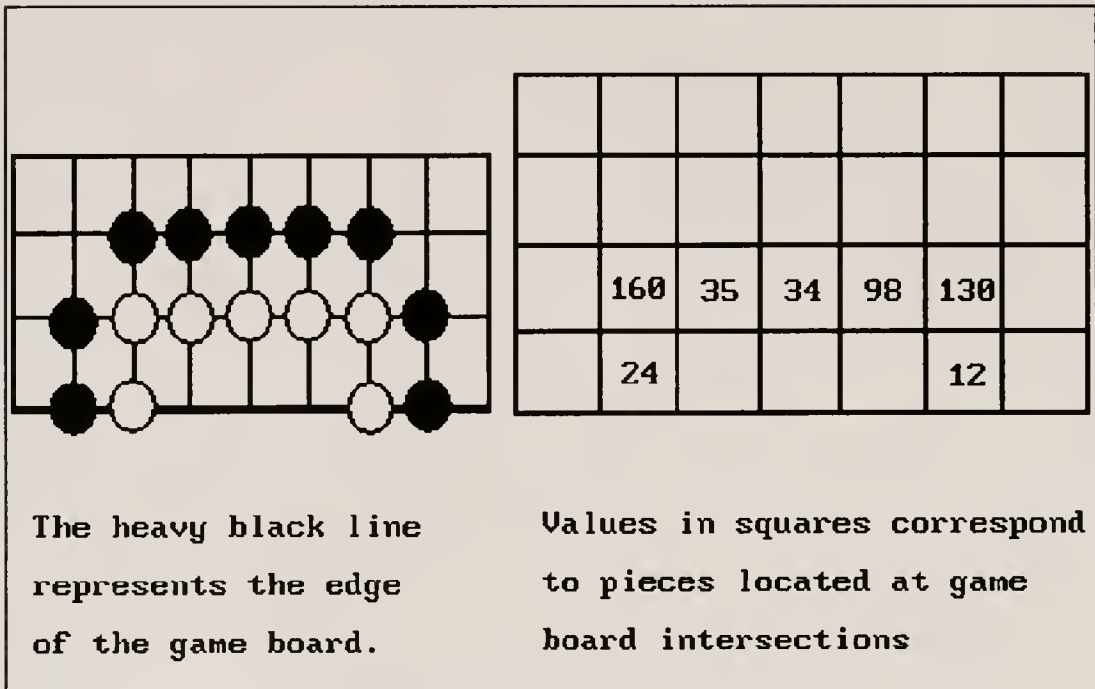
Figure 14: Chunk acquisition in Go.

to acquire textual chunks. Each move from a previously played game is read in textually, in modified algebraic notation, from the game record used as input. The algebraic notation is then translated to update the current board representation maintained by the adversary modeling methodology application. Prior to translating the textual notation used to record the moves of an adversary's game, specific types of moves are saved in the "move" section of the adversary knowledge base verbatim.

Due to the availability of transpositions which lead to the same board position through several different move sequences, we store each of the first five moves performed by an adversary individually. A statistical representation of the frequency that a move occurs and the average game turn is generated and stored with each move. The statistical representation includes: the number of times a specific move

has been observed, the mean move number at which the move has occurred, and the color of the pieces being played.

From the statistical knowledge of each of the moves displayed by the adversary, the adversary model predicts opening sequences of play based on the average over-all performance of an adversary. The knowledge of piece color is crucial because chess masters use different modes of play for each of the two colors. As the white player, a chess master can choose which opening sequence to play and has greater control over the position of the pieces following the opening. However, as the black player, a chess master plays a responsive mode instead of the controlling mode of white. The opening moves of black are made as responses to the white player's move choices. Therefore, if an identical move is made by an adversary as both a white piece and black piece player, then the two moves are considered as different moves and are stored separately.

Other types of moves are useful for predicting the playing style of an adversary. We save the number of pawn moves that occur during the first ten moves of each game and the number of attacking moves made during the first ten and twenty moves of each game to aid in inferring the playing style of an adversary. Any other move which can be identified textually is also a candidate for the textual move knowledge of the adversary model. A flowchart for the process of collecting textual-based knowledge is shown in Figure 15. The frequency and direction of castling moves are also acquired to assist in prediction of possible castle moves during an adversary's future games.
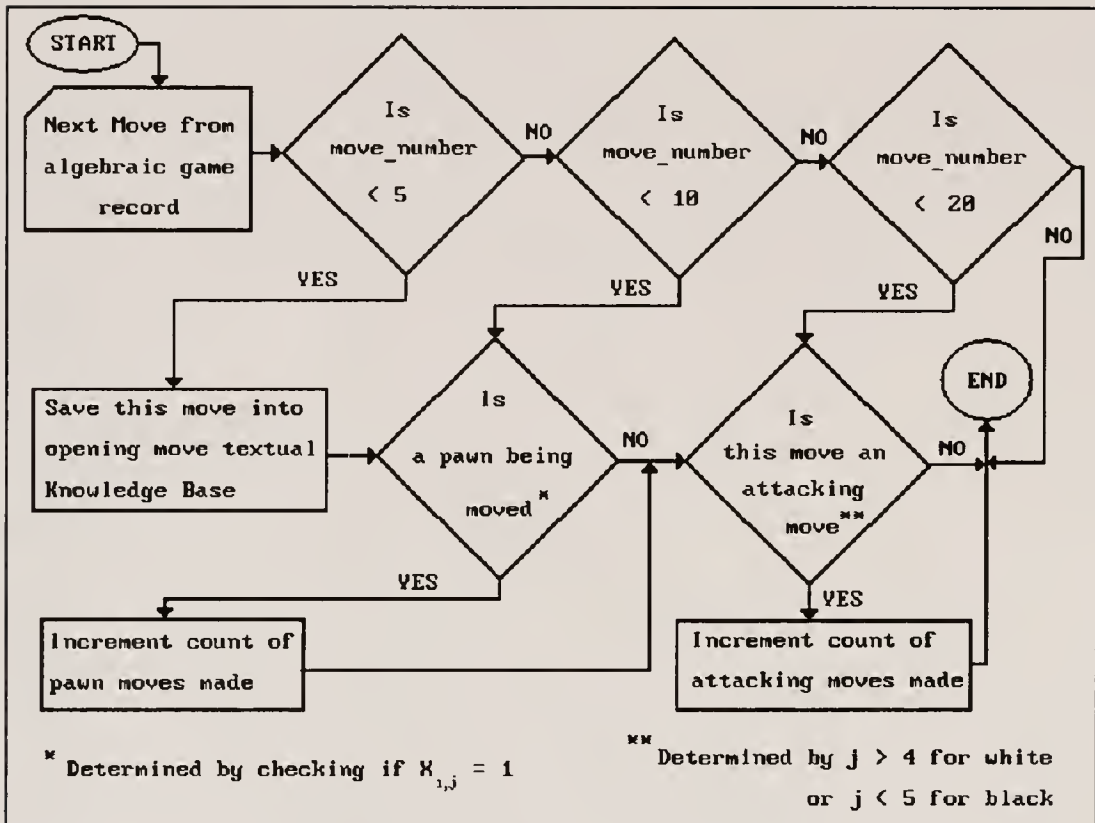
Figure 15: Flowchart to collect textual move knowledge.

To demonstrate the flowchart, assume that the move c4--pawn to queen's bishop four--is made by the adversary on his fourth game turn. Since the move occurs during the first five moves of the game, the move will be saved into the textual chunk knowledge base. If the move already exists in the textual chunk knowledge base, then the number of times this move has been observed is incremented and the statistics regarding the time of the move are updated to reflect the new knowledge. Because a pawn--the queen's bishop pawn--is being moved, we increment the count of pawn moves made during the first ten moves of the game.

We save the total number of games analyzed, the number of games the adversary played as white, the number of games won by the adversary, the number

of games won as white, and the average length of all analyzed games. This information is used by the adversary model to determine the frequency of a specific move and the likelihood that the move will be repeated by the adversary.

Other adversarial domains can benefit from the textual knowledge acquisition performed by the adversary modeling methodology. For example, the acquisition of textual patterns will capture standard operating procedures and protocols, such as how to accept an enemy's surrender, that have been learned by an adversary in a military domain.

3.2 Induction to Learn Repeated Chunk Patterns

We have acquired the geometric chunks, both standard and pawn structure chunks, through the computer vision technique of applying convolutions to the board image. Our goal with the adversary modeling methodology is to learn the chunks which are used by the adversary to evaluate domain situations. We propose that the chunks which are commonly used by an adversary can be identified through their repeated use across several games.

Induction is used to identify groups of chunks from the collection of chunks acquired from each of the analyzed games. Each group which contains a minimum of two chunks, indicating the observation of the chunk in two separate games, is identified as a cognitive chunk used by the adversary in domain evaluations. Chunks that are found only in single games may occur by chance and are therefore not used to predict the tactical and strategic decisions of an adversary. Because of our

inductive premise, we require that a minimum of two adversary games be studied prior to performing the induction.

The induction algorithm requires chunks to be identical before being grouped together. Our induction algorithm is noise intolerant since even small variations in a chunk can have a large effect on the strategic evaluation value of a chunk. An example of three chunks which only differ by the location or value of a single piece and which have significantly different strategic meanings is shown in Figure 16. The first chunk shows a strong pawn formation, while the second chunk shows two pawns being attacked by an opponent's pawn, and the third chunk shows a pawn formation with a double pawn which may become a liability towards the end of the game. The noise intolerance of the induction algorithm is not a real problem since most adversarial domains keep very accurate records of the past performances of domain entities.
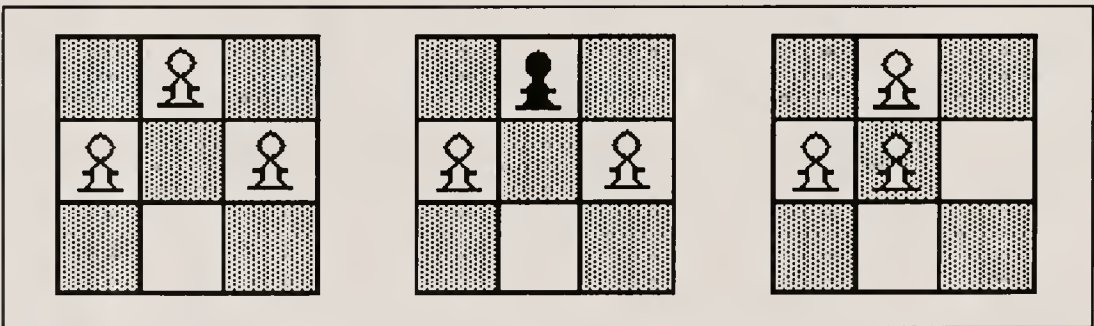
Figure 16: The effect of noise on chunks.

Chunks are considered identical when each has the same number and type of pieces arranged in the same geometric configurations. The strategic significance of a chunk is usually invariant with respect to left and right alignment, so chunks are reflected through a vertical axis of the game board while trying to identify identical

chunks. When the adversary is playing the black pieces, we reflect the pieces through the mid-board horizontal axis to group chunks that occur while the adversary is playing either color of pieces. Chunks considered to be identical, as shown in Figure 17, are grouped together and stored in the knowledge base that forms the adversary model.
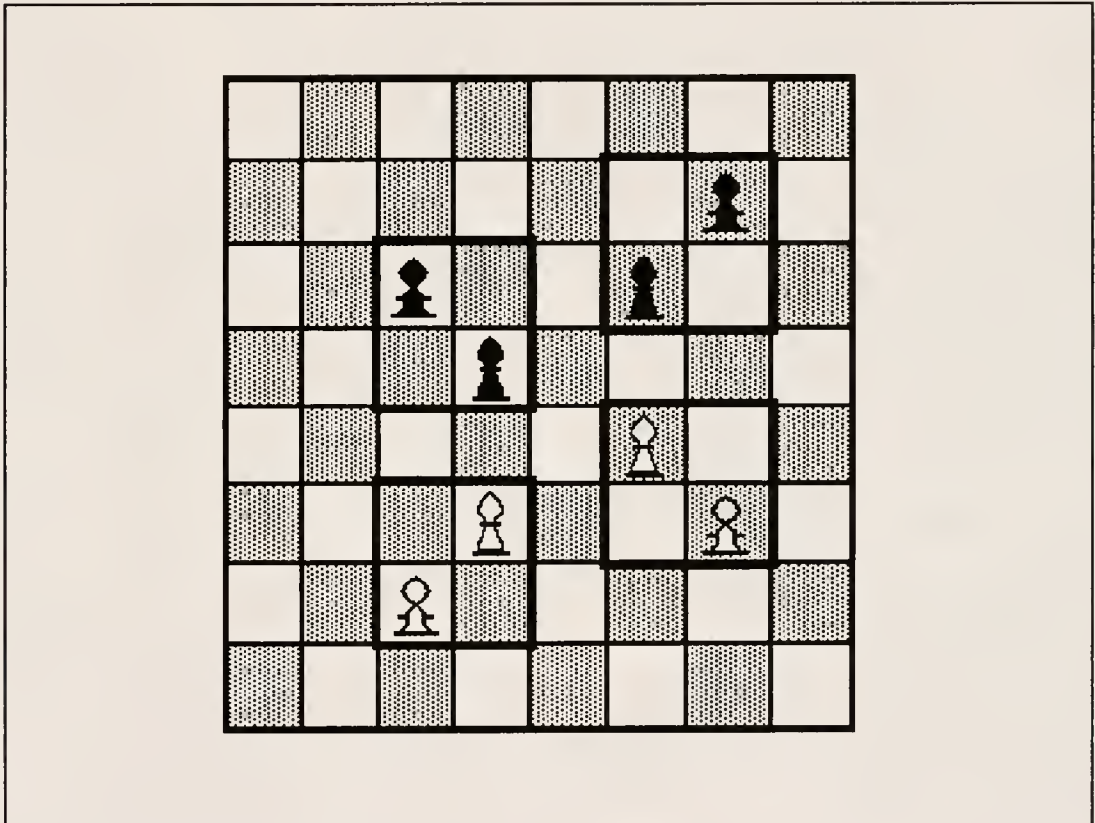
Figure 17: Four identical chunks.

We use a frame based system to store the chunks found by the induction algorithm in the adversary model's knowledge base. The slots in each frame are the actual chunk pieces and relative positions of the induced chunks, the number of times this chunk has been observed, the earliest move number when the chunk was created, the latest move number when the chunk was dissolved, the outcome (win or loss) of

the games in which the chunk occurred, and the color of the adversary's pieces. Although we have not implemented this slot, chunks which are observed frequently can have a "best move" slot added which would model the scripts used by chess masters for certain configurations.

The adversary model's knowledge base of patterns known to an adversary can be constructed in parts. At least two games need to be processed by the convolution operator to acquire potential chunks. After the potential chunks have been acquired, the induction learning mechanism processes the potential chunks of each game by first checking to see if the chunk already exists in the adversary model's knowledge base. If the chunk already exists in the knowledge base then the appropriate slots are updated to reflect the new information associated with the new potential chunk. Should the chunk not exist in the current knowledge base, then the chunk is compared against the chunks of every other game that has just been processed by the convolution operation. Any identical chunks are grouped and stored in the knowledge base as a new pattern known to the adversary.

The induction learning mechanism is applied separately to general chunks containing different pieces of the same color and pawn structure chunks containing only pawns of both colors. Following the induction procedure, the adversary model has a knowledge base which identifies all the recurring textual and geometric patterns from the previous games of a particular adversary. Multiple adversaries can be modelled by the adversary modeling methodology with each adversary having a separate knowledge base. The adversary model's knowledge bases of specific

adversaries are used to infer the playing style of an adversary and to predict the tactical and strategic movement choices of an adversary.

3.3 Acquiring an Adversary's Playing Style

Knowing the general playing style of an adversary enables chess programs to select moves that will place the adversary in a disadvantageous strategic position. The general playing style of an adversary is inferred from the collection of patterns contained in the adversary model knowledge base. We use a rule-based inference engine to heuristically determine the playing style of an adversary.

Currently, we infer two different styles of play. The inference engine indicates that an adversary prefers open positions, closed positions, or has no obvious preference. The closed position style of play is further used to imply strong positional play by the adversary. Likewise, an open position playing style implies a strong tactical player.

The rule base heuristics address the following domain specific factors which indicate an adversary's playing style preference:

- The presence and relative size of pawn structures.

- The number of pawn moves made during the opening sequence.

- The presence of pawn chains, or inter-locked pawn structures.

- The number of opening moves that are attacking--move a piece across the mid-board horizontal axis.

Each rule is evaluated to determine if a specific playing style is demonstrated. The

rules are not competitive, but instead they lend support to the adversary model's belief that a particular playing style is preferred by the adversary by adjusting the likelihood value attached to each playing style. The playing style with the greatest likelihood is then inferred as the general playing style of the adversary.
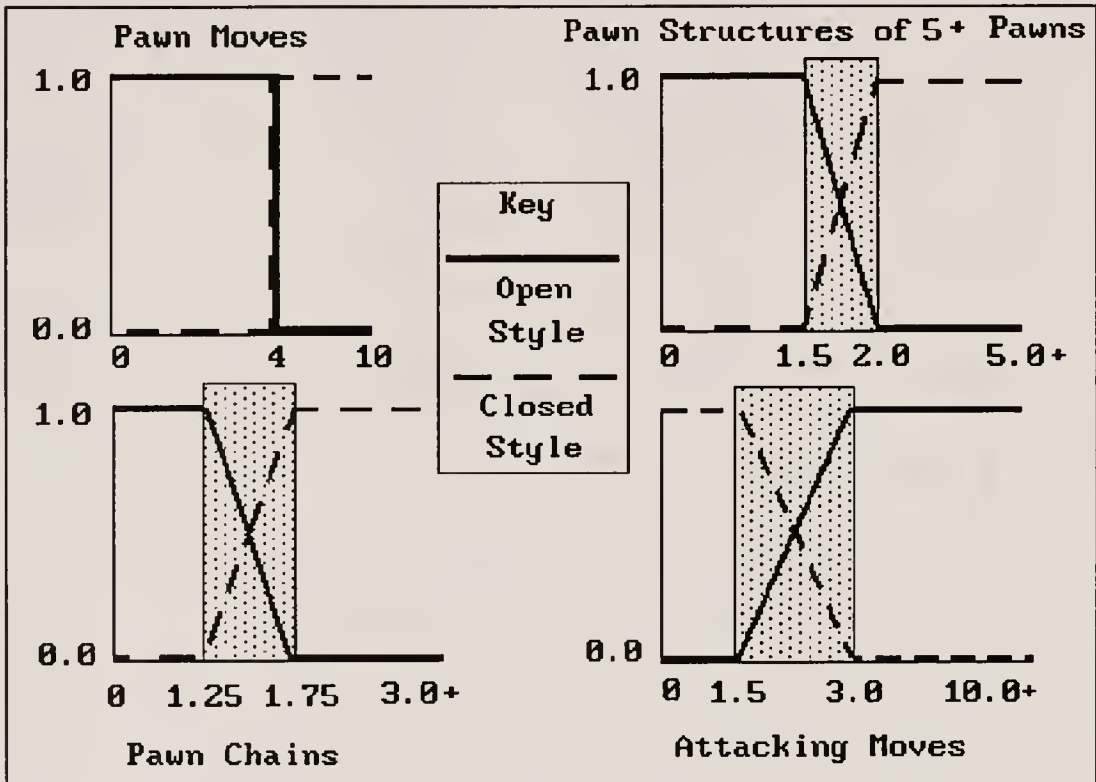
Figure 18: Fuzzy logic used by the heuristic rules.

The heuristic rules use fuzzy logic during the evaluation process. Graphs of the fuzzy values for each of the heuristic rules is shown in Figure 18, with the fuzzy regions shaded. Heuristic rules that have a value in the fuzzy areas which lie close to the division between the two styles of play currently inferred do not support either hypothesis. Our decision to use fuzzy logic prevents us from assigning a playing style to an adversary that is based on statistically insignificant data with respect to the fuzzy set values.

The heuristic rules used to infer an adversary's playing style are not competitive, new heuristic rules can be added at any location of the rule base. As new domain specific knowledge becomes available that will infer additional playing styles or support the current playing style inferences, the current rule base can be easily augmented.

The determination of an adversary's playing style preference has been added to the generalized adversary modeling methodology to increase the adversary model's performance in the chess domain. The playing style inference engine makes use of domain specific knowledge in addition to the textual and geometric patterns and for this reason is not considered as a main part of the adversary modeling methodology. The pattern acquisition, induction learning mechanism, and adversary action prediction tool each use only minimal amounts of domain specific knowledge so the adversary model implementation can be used in other adversarial domains with little or no modifications.

3.4 Application of the Adversary Model

The adversary model consists of a knowledge base of textual and geometric, or visual, patterns known to the adversary. The third part of our hypothesis presented at the beginning of this chapter indicates how we are going to use this knowledge. An adversary will attempt to reduce the complexity of evaluating the current domain situation by maneuvering the board position to contain one or more of the pattern chunks with which he is familiar.

The adversary model also contains two processes which utilize the knowledge about patterns known by an adversary. The first process detects analogous situations in the domain and the second process attempts to predict the most probable adversary action. We define an analogous domain situation to exist when a chunk in the knowledge base is almost complete in the domain. The second process then predicts the necessary move to complete the chunk.

Currently a move window size of one is used when detecting analogous domain situations. For chess, this means that the chunk must be able to have all of the pieces of the chunk in their correct relational positions in exactly one move. Board positions that are considered to be analogous to chunks in the knowledge base may only have at most two pieces out of position, but are normally required to only have one piece out of position. An extension to the move window size has been implemented to permit the adversary model to eliminate interfering pieces when a castle is the required move to complete a chunk.

We detect the analogous situations we have defined for the chess domain by performing pattern recognition on the game board. The board representation of the adversary's game currently in progress is searched prior to each move to be made by the adversary until a piece belonging to the adversary is located. Next all of the chunks in the adversary model's knowledge base which can fit into the current board location are compared against the current board pattern. Inconsistencies between the chunks in the knowledge base and the current board configuration are counted. An inconsistency exists if a board location corresponding to the location of a piece

in a knowledge base chunk is empty or contains a different piece. If only one inconsistency exists, or two if the missing pieces are the king and rook, then the analogous situation detection process checks to see if any viable move can complete the chunk.

Chunks in the knowledge base occupy a square area of the game board defined by the pieces farthest to the left, top, right, and bottom of the chunk. The square areas of chunks frequently contain blank or non-occupied squares. These squares serve as wild cards so that the actual square on the game board may be blank or contain any other piece. This is done to simulate the effect of hierarchically composing larger chunks from smaller chunks.

Each chunk in the knowledge base which can be completed suggests the appropriate move to complete the chunk as the adversary's next move choice. Several chunks can each have analogous situations for any particular turn of the current game. The collection of possible moves that enable an adversary to complete a chunk are evaluated by the adversary move prediction process to select the most likely adversary move.

The adversary move prediction process uses a heuristic rule-based inference engine to evaluate the collection of suggested moves that will complete chunks. The inference engine is similar to the one described above for analyzing the playing style of an adversary. An initial probability value is generated for each move by dividing the number of times the chunk suggesting the move has been observed by the total number of prior adversary games analyzed. The heuristic rules then add or subtract

from the base probability of each suggested move. The move with the highest probability is then used to predict the adversary's next move. If two or more moves have nearly equal probabilities, then the inference engine makes multiple move predictions.

Computer chess programs can use the probability value associated with each move as a measure of belief. When the adversary model infers a probability for a move that is less than some specified value, the chess program can choose to ignore the adversary model's prediction.

The heuristics used in the prediction inference engine attempt to follow several psychological principles of cognitive organization and economy. The general description of the purpose of our heuristics follows.

- Large chunks are favored over small chunks. Chunk size is dependent on the number of pieces contained in a chunk.

- Chunks containing major pieces are favored over chunks containing only pawns.

- Favor moves which have been suggested by more than one chunk.

- Reduce the probability of chunks that have only been observed in lost games.

- Increase the probability of chunks whose move suggestion causes a gain in material advantage. (This heuristic is only used in quiescent positions due to the volatile nature of material advantage in non-quiescent positions.)

- Chunks which have occurred while using the same color pieces as the current game are favored over chunks that have only been observed for the opposite color.

- Adjust the probability to account for temporal discrepancies. If the current game move number is outside of the move range defined by the two slots for the chunk's time of inception and dissolution, then reduce the probability proportional to the distance between the current move and the chunk's temporal range.

- Eliminate smaller chunks that attempt to borrow pieces from an existing larger chunk.

- Reduce the probability of chunks containing only two pieces during the middle game segment.

- If the adversary has just dissolved a chunk, then do not recreate the chunk immediately.

- Reduce the probability of chunks that have a maximum dissolution game turn that is within the time range covered by the textual opening move knowledge base.

These heuristics attempt to find the move suggestion which has the greatest number of situational factors in common with the current game condition, such as the color of the pieces and the time or game turn within the game. Additionally, the heuristics promote the moves that will create the largest possible chunk and therefore afford the greatest cognitive economy to the chess player. The heuristic which supports moves suggested by multiple chunks is simulating the construction of hierarchically complex chunks from smaller chunks. Our general description of the heuristic rules indicates a minimal usage of domain specific knowledge which augments our inter-domain application capabilities.

The role of the adversary model as a coach needs to be remembered. Predictions of probable adversary actions are only made when the current game situation resembles domain situations which have already been learned inductively. After analyzing nine and then twelve games of Botvinnik the resultant adversary models were used to predict the actions of Botvinnik in a new game which had not been previously analyzed. For each of models' predictions, over forty percent of the new game's board configurations had no similarities to the chunks in the knowledge base.

Geometric chunk patterns are uncommon during the opening game segment of a chess game. The textual knowledge base of patterns known to the adversary is used to supplement the lack of knowledge in the geometric chunk pattern knowledge base. When a computer chess program equipped with the adversary model is playing the white pieces, the statistical analysis of opening moves displayed by the adversary is used to select an opening sequence of play which is unfamiliar to the adversary. This grants a strategic advantage to the computer chess program with the first move of the game. Predictions for the opening moves of the adversary are made by selecting the move with the highest probability value. Probability values are assigned by using the statistical values stored with each opening move displayed by the adversary to attain a base probability corresponding to the frequency each move has been observed. This base probability is then modified to account for any difference between the current time of the game and the mean game turn of the knowledge base textual move pattern.

Moves which have already been made in the current game cause the identical move in the knowledge base to be rejected. Additionally, when the adversary is playing the black pieces, we implemented "response code" in the prediction process which heuristically favors textual move patterns that have previously followed the current move just made by the white player. The effect of the response code is examined in Chapter 4.

CHAPTER 4
IAM

In this chapter we present a detailed examination of our implementation of our adversary modeling methodology for the domain of chess. Our program is named IAM, an acronym for Inductive Adversary Modeler. We first present several detailed examples that demonstrate IAM's functionality. Next we review the effectiveness of IAM by examining the predictive capabilities of our adversary model while performing against an actual opponent.

4.1 Overview of IAM's Execution

IAM is defined in two stages. The first stage performs the knowledge acquisition of chunks and the learning by induction phases of the adversary modeling methodology which establish the knowledge base of geometric and textual chunks contained in the adversary model. The second stage applies the knowledge in the knowledge base to a game in progress to predict probable adversary moves and to identify the adversary's playing style.

IAM has several global data structures which facilitate the algorithmic design of the program. Chunks, both textual and geometric, and the suggested moves of the adversary model are stored in dynamic data structures. The form of these structures and the pointers to them are among the global data structures. Additionally,

application constants such as the size of geometric chunks and pawn structure chunks, the number of opening moves to acquire, the internal representation of the chess board, and various constants used by the heuristics of the two inference engines are stored as global variables. The use of these global structures allows us to easily change the amount, size, or type of knowledge that is to be acquired and provides a means for fine tuning the heuristics used in predicting an adversary's move choices.

4.1.1 Knowledge Acquisition and Inductive Learning Stage

Input to IAM, for the knowledge acquisition and learning stage, consists of two or three game records of the adversary's previous performances. The records are written in algebraic notation as shown for the sample game in Figure 19. The first line of the game record indicates the color of pieces played by the adversary, the length of the game, and the game result, either win or loss, for the adversary. The remaining lines are the actual moves of a particular game.

We use only the previous games of an adversary which have resulted in a win or loss. Drawn games are often influenced by factors, like the state of health of the adversary, that do not impart any knowledge about the strategic tendencies or evaluation criteria of an adversary. A good example of this comes from the 1957 World Chess Championship match between Botvinnik and Smyslov. Three of the last four games in the match ended in a draw with an average game length of thirteen turns. Botvinnik was down by three points in this match and the first draw was probably offered by Botvinnik so that he could rest and prepare for the next game. The final two draws were offered and accepted because the result of the match was

Botvinnik vs. Keres (Moscow, May 4/5, 1948)

1 60 1

1.	d4	d5	21.	Nb4	Rd8	41.	Kd3	Ne6
2.	Nf3	Bf5	22.	Qf5	Rd6	42.	Nd5	Kc6
3.	c4	e6	23.	Rfc1	Rxc6	43.	h4	Nd8
4.	cxd5	exd5	24.	Rxc6	Rd8	44.	Nf4	Kd6
5.	Qb3	Nc6	25.	Rxb6	cxb6	45.	Nh5	Ne6
6.	Bg5	Be7	26.	Nc6	Qc7	46.	Ke3	Ke7
7.	Bxe7	Ngxe7	27.	Nxd8	Qxd8	47.	d5	Nc5
8.	e3	Qd6	28.	Qc2	Qc7	48.	Nxg7	Kd6
9.	Nbd2	O-O	29.	Qxc7	Nxc7	49.	Ne6	Nd7
10.	Rc1	a5	30.	Nb1	Kf8	50.	Kd4	Ne5
11.	a3	Rfc8	31.	Kf1	Ke7	51.	Ng7	Nc4
12.	Bd3	a4	32.	Ke2	Kd6	52.	Nf5+	Kc7
13.	Qc2	Bxd3	33.	Kd3	Kc6	53.	Kc3	Kd7
14.	Qxd3	Nd8	34.	Nc3	Ne8	54.	g4	Ne5
15.	O-O	Ne6	35.	Na2	f6	55.	g5	fxg5
16.	Rc3	b5	36.	f3	Nc7	56.	hxg5	Nf3
17.	Qc2	Rcb8	37.	Nb4+	Kd6	57.	Kb4	Nxg5
18.	Ne1	Nc8	38.	e4	dxe4+	58.	e5	h5
19.	Rc6	Qe7	39.	fxe4	Ne6	59.	e6+	Kd8
20.	Nd3	Nb6	40.	Ke3	Nc7	60.	Kxb5	

1:0

Figure 19: An example of a game record--the 1948 Hague-Moscow Tournament.

a forgone conclusion, similar to our example in Chapter 2 of the young Soviet chess master and the older Soviet grandmaster.

Additional knowledge can be inferred from the win/loss result of each game played by an adversary. When the induction procedure detects a series of lost games by the adversary, then the possibility that the adversary will be altering his established strategy is inferred. The adversary move prediction inference engine decreases the probability of chunks observed only in lost games from predicting the next adversary move. Likewise, an overall winning performance by an adversary reinforces the likelihood of chunks being repeated in future games.

Each move of every game is read in as input separately. The textual chunking mechanism then identifies if the current move has any of the qualifications for being saved in the textual portion of the knowledge base. If so, a dynamic data structure for holding textual moves is created and the move is stored.

Next, the textual move is translated and the internal board representation is updated. After each move made by the adversary, potential chunks for each game are identified by the convolution operator we discussed in Chapter 3. Chunks which are identical to another chunk in the same game are coalesced into a single chunk (see the pseudo-code in Figure 12). This step, for the twelve games played by Botvinnik in the 1948 Hague-Moscow Championship tournament, reduced the 699 chunks identified by the convolution operator to 272 potential chunks that the inductive learning algorithm must process. Chunks from different games are considered unique at this point in processing.

Pawn structure chunks are collected by the convolution operator separately from the standard geometric piece chunks. Because the occurrence of identical pawn structures during different segments of a game yields knowledge about an adversary's playing style, pawn structure chunks are not coalesced.

After the individual geometric chunks have been collected for each game, the induction algorithm processes the chunks to find patterns which are duplicated across game boundaries. The induction algorithm found twenty-eight unique chunks, shown in Figure 6, from the 272 potential chunks identified for Botvinnik in the Hague-Moscow tournament. Each of the twenty-eight chunks induced was found in

at least two games. For example, the partial board representations for the three games which produced one of the twenty-eight chunks are shown in Figure 20.
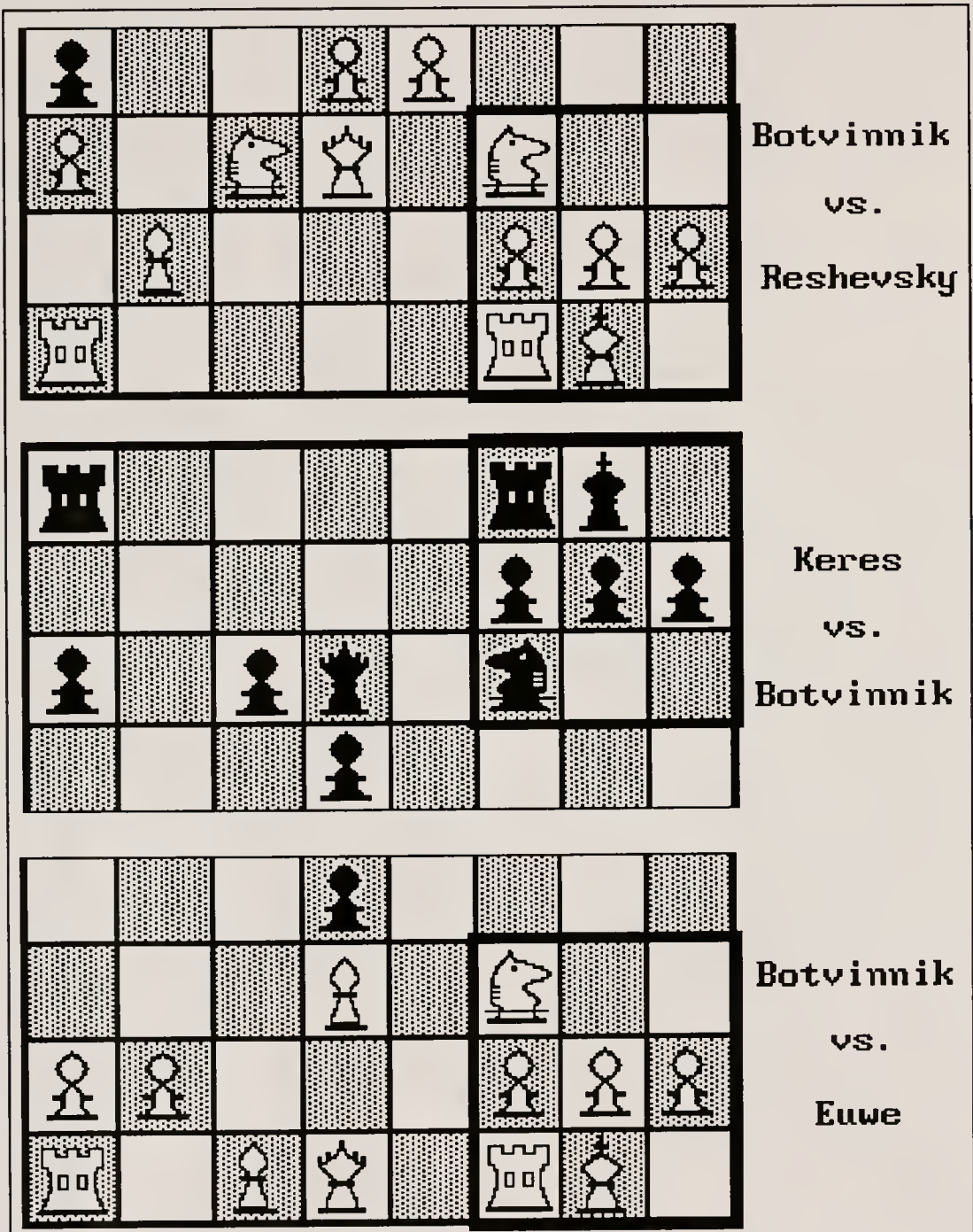
Figure 20: Chunk occurrence in three different games.

The induction algorithm, with pseudo-code shown in Figure 21, verifies if a potential or induced chunk already exists in the adversary pattern knowledge base and updates the knowledge base appropriately. When a chunk is found to be repeated in multiple games, we use the function REMOVE to eliminate multiple instances of the chunk from the list of potential chunks produced from our convolution algorithm. The induced chunks which do not exist in the current knowledge base are then written into new frames, created by the function NEW, in the knowledge base by the function WRITE.

The potential chunks collected by the convolution operator which are not grouped into a valid chunk to be saved in the knowledge base are written to a temporary file. This temporary file serves as a database of non-repeating chunks which is used to identify chunk repetitions that occur outside of the current games being induced. All chunks in this temporary database are used by the induction algorithm to group chunks that have not been previously grouped. The database is deleted after all games from a specific time period or tournament have been analyzed. The deletion of the database forces the induced chunks to have a temporal relevance, or recency, to the potential chunks being analyzed.

The textual move chunks are then analyzed to gather the statistics to be used by the prediction stage of IAM. The statistics that are gathered include the mean game turn that a move was executed, the number of times a particular move was observed, the results of the games in which the move was observed, and the color of the pieces being played. This statistical analysis requires that only one entry be made

```
For I = 1 to number of chunks found for the current games  do
   Begin
      Flag_1 ← FALSE     ;; indicates when a chunk has been repeated
      Flag_2 ← FALSE     ;; indicates when a chunk is duplicated in the KB
      For J = I+1 to number of chunks found for the current games  do
      ;; First check this chunk against the chunks that were just found
         If C_I = C_J
            Then Begin
                    Increment number of times C_I has been observed
                    If C_I.start_time > C_J.start_time
                    Then C_I.start_time ← C_J.start_time
                    If C_I.end_time < C_J.end_time
                    Then C_I.end_time ← C_J.end_time
                    REMOVE(C_J)
                    Flag_1 ← TRUE
                 End Begin
      For J = 1 to number of chunks currently in the chunk Knowledge Base do
      ;; Next check this chunk against all chunks already in the Knowledge Base
         If C_I = C_J
            Then Begin                          ;; Update KB knowledge
                    Increment number of times C_J has been observed
                    If C_I.start_time < C_J.start_time
                    Then C_J.start_time ← C_I.start_time
                    If C_I.end_time > C_J.end_time
                    Then C_J.end_time ← C_I.end_time
                    Flag_2 ← TRUE
                 End Begin
      If Flag_1 = TRUE AND Flag_2 = FALSE
         Then Begin
                 NEW(geometric_frame, F)
                 F ← C_I
                 WRITE(F, Geometric_Chunk_Knowledge_Base)
              End Begin
   End Begin
```

Figure 21: Pseudo-code to perform induction on geometric chunks.

in the knowledge base for each specific move. Additional information concerning the total number of games analyzed, number of games played as white, number of games won, and number of games won as white is stored with the textual move patterns.

The pawn structure chunks are also analyzed statistically at the same time as the textual move patterns. The statistical analysis gathers knowledge that is used by the adversary playing style inference engine. This knowledge is about the presence, quantity, and quality of pawn structures and inter-locked pawn chains. The quality of a pawn structure is determined by the number of pawns involved in the structure. The knowledge from this analysis is stored with the textual moves in the adversary patterns knowledge base.

Following the induction on geometric chunks and the statistical analysis of pawn structures and rote learned textual chunks, the knowledge base of patterns known by an adversary is available for use by the prediction stage of the adversary model. The adversary pattern knowledge base is constructed incrementally. As new games of an adversary become available the chunk knowledge from those games can be quickly incorporated into the existing knowledge base.

4.1.2 Predicting Adversary Actions Stage

After an adversary patterns knowledge base has been constructed, the adversary model then attempts to predict adversary actions in relevant domain situations. Simulations of a game played by an adversary are used to verify the prediction capabilities of the adversary modeling methodology. Each of the simulated games used to represent a future contest against an adversary has never been analyzed by IAM.

Prior to each of these simulated games, the preferred playing style of the adversary is inferred from the chunks in the knowledge base. We had IAM study twelve games of two different chess grandmasters who display preferences for the two different playing styles currently inferred by IAM. The two players were Botvinnik, who prefers closed positions, and Spassky, who prefers open positions. By analyzing the presence and frequency of pawn structures and the frequency of pawn and attacking moves, IAM correctly identified the playing style preference of each adversary. Knowledge about the playing style of an adversary is then used by the current domain program which the adversary model is coaching to select specific moves that manipulate the game board away from the adversary's preferred style.

During the opening portion of the simulated game, IAM relies exclusively on the textual move knowledge base which contains the statistical analysis of opening patterns displayed by the adversary. Prior to the adversary's move, IAM predicts the adversary's move from the moves currently in the knowledge base. Following the actual move made by the adversary, the corresponding move in the knowledge base is eliminated from further consideration. For an adversary playing the white pieces, IAM has had accurate predictions ranging from forty percent to one hundred percent, with a mean prediction accuracy of seventy-five percent.

IAM's opening sequence predictions are less accurate for an adversary playing the black pieces. This is because the adversary is playing responsively to his opponent's moves. Accuracy for opening move predictions against a black piece playing adversary ranged from zero to twenty percent.

By modifying the textual chunk learning algorithm to remember the white player's move which preceded an adversary's black move, we were able to significantly increase IAM's opening sequence prediction accuracy to range between forty and one hundred percent, with a mean of sixty percent. This modification produces results which are similar to IAM's performance against the adversary while playing the white pieces.

The number of textual chunks stored in the knowledge base increases proportionately, since identical moves by the adversary which follow a different white move are now considered as unique moves. The average increase in the knowledge base size is forty-five percent. For example, after analyzing twenty-two games of Botvinnik with the original textual chunking algorithm, thirty-one opening move textual chunks were produced. With the response code modification forty-seven opening move textual chunks were saved to the knowledge base. The actual textual chunks are also larger to contain the adversary's opponent's move. Because of the relatively small number of chunks in the knowledge base for a specific opponent, the increase in prediction accuracy warrants the small increase in size to the knowledge base.

All move predictions following the opening sequence are made primarily from the contents of the geometric piece chunk knowledge base. Before each move to be made by the adversary, the current board representation is searched to find pieces belonging to the adversary. This is done because each of the chunks contained in the knowledge base consists of pieces belonging to the adversary.

Once an adversary's piece has been located on the board representation, each chunk in the knowledge base is compared against the board representation to find identical matches and analogous chunks, which have one or two pieces out of place. Because chunks occupy a specific square area of the game board, chunks which are to large to fit into the current board location are ignored to reduce processing time. The detection of analogous board positions is performed via pattern matching. When an analogous board position has been identified, IAM then checks to see if a legal chess move would complete the analogous chunk. An example of this process is shown in Figure 22. Four of the chunks shown in Figure 6 were used to identify analogous positions on the current game board which is prior to Botvinnik's thirteenth move from the game displayed in Figure 19. The corresponding moves that would produce the chunks are also displayed in the figure.

The collection of moves suggested by the analogous chunks is then processed by an inference engine to heuristically determine the most probable move. The inference engine predicted the castling move for the Figure 22 board position. This prediction represents the result of applying the various psychologically based heuristics concerning chunk size and relevance to the current domain. The queen being en prise is domain specific information and is therefore not considered by the move prediction inference engine. However, the actual move made by Botvinnik, Qc2 or b3-c2, is one of the moves suggested by an analogous chunk and reviewed by the inference engine. An experiment to augment the adversary pattern knowledge base with a minimal amount of domain specific knowledge is presented in section 4.2.
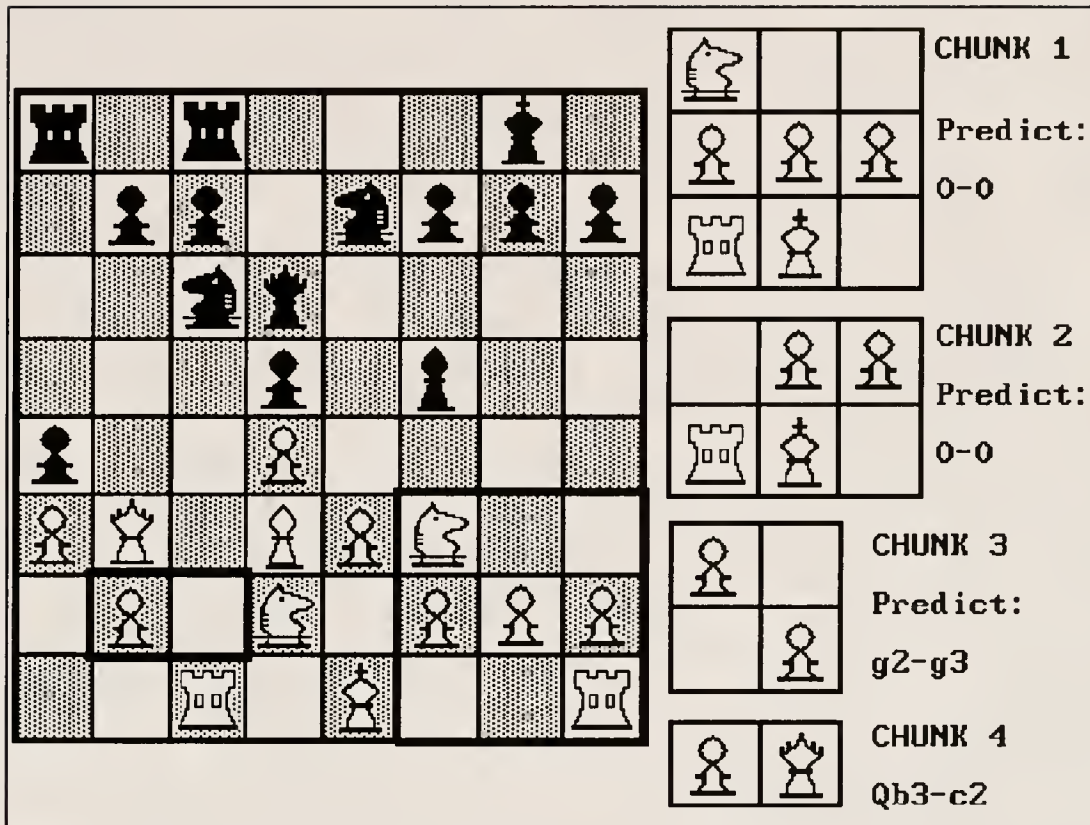
Figure 22: Predictions from four chunks.

The process of searching the current board configuration to identify analogous positions is repeated before each move to be made by the adversary. Existing chess programs can use IAM's predictive capabilities by having IAM search the possible board configuration at a specific node of the game tree and finding analogous chunk situations in the proposed board configuration. IAM will then predict the most probable adversary move that corresponds to our domain simplification hypothesis of Chapter 3.

4.1.3 Integrating IAM's Knowledge into a Chess Program

Our induction based adversary modeling methodology produces knowledge which can be incorporated into current chess programs by two methods. Knowledge

about the specific opening sequences preferred by an adversary can be used to manually optimize a chess program's opening book. Optimization of the opening book entails augmenting the opening book with opening sequences that are unknown to the adversary and adjusting the preferences of the chess program to select these new opening sequences.

Current chess programs use a move generator to limit the breadth of their game trees. Move generators suggest a number of moves, typically five, which are considered as the only moves available from a specific position. We describe the reduction of tree complexity by the adversary modeling methodology in Chapter 5. IAM predicts the move that an adversary will execute next from a particular board position. These predictions can be used by chess programs to further limit the search of their game trees.

A chess program would have the adversary model make predictions of the next move in the game tree while simultaneously generating the five best moves with its move generator. When predictions from the adversary model are available, the chess program would use the predictions to expand its current game tree. Otherwise, the chess program would continue to use the moves supplied by its move generator. Combinations of adversary model predicted moves and move generator suggested moves can be used by heuristically ordering the moves suggested by both methods and expanding the game tree with the first five moves from the ordered move list.

4.2 IAM's Performance Results

In this section we describe several experiments that have been performed with IAM to test the overall capabilities of the adversary modeling methodology. Each experiment is described preceding the analysis of the results.

The games of the adversary Botvinnik are all taken from his World Championship matches from the 1948 Hague-Moscow tournament through his 1961 match against Tal. The two games which are used to simulate Botvinnik's performance in a future match, come from Botvinnik's 1963 match against Petrosian, and are shown in Figure 23 and Figure 24. The game TEST1 is the first game from the Petrosian match resulting in a win or loss in which Botvinnik played the white pieces. TEST2 is the next game from the match, following TEST1, in which Botvinnik played the black pieces.

4.2.1 General Performance Results

The purpose of our first experiment is to determine the base performance results of IAM so that we can compare these results against other experimental results. We anticipate that the predictive capabilities of IAM will increase as more chunks are learned. The records of every game played by Botvinnik from his World Championship Matches, prior to 1963, are used as input.

4.2.1.1 Performance versus the adversary as the white player

TEST1 is simulated for IAM following the construction of the adversary model's knowledge base for each of the tournaments of Botvinnik's career. The

length in game turns of TEST1 is fifty-seven moves. The performance of IAM in this experiment is shown in Table 1.

TEST1--Botvinnik playing White											
1.	d4	d5	21.	Re2	Nb6	41.	Nc5	Bf5			
2.	c4	e6	22.	Rhe1	Nc4	42.	Kg3	a4			
3.	Nc3	Be7	23.	Bxc4	Rxc4	43.	Kf4	a3			
4.	cxd5	exd5	24.	Rd2	Re8	44.	Ke5	Rb4			
5.	Bf4	c6	25.	Re3	a6	45.	Nd3	Rb5			
6.	e3	Bf5	26.	b3	Rc6	46.	Kd6	Kf7			
7.	g4	Be6	27.	Na4	b6	47.	Kc6	Bxd3			
8.	h3	Nf6	28.	Nb2	a5	48.	Rxd3	Rb2			
9.	Bd3	c5	29.	Nd3	f6	49.	Rxa3	Rg2			
10.	Nf3	Nc6	30.	h4	Bf7	50.	Kxd5	Rxg5+			
11.	Kf1	O-O	31.	Rxe8+	Be6	51.	Kc6	h5			
12.	Kg2	cxd4	32.	Qe3	Bf7	52.	d5	Rg2			
13.	Nxd4	Nxd4	33.	g5	Be6	53.	d6	Rc2+			
14.	exd4	Nd7	34.	Nf4	Bf7	54.	Kd7	h4			
15.	Qc2	Nf6	35.	Nd3	Be6	55.	f4	Rf2			
16.	f3	Rc8	36.	gxf6	Qxf6	56.	Kc8	Rxf4			
17.	Be5	Bd6	37.	Qg5	Qxg5+	57.	Ra7+				
18.	Rae1	Bxe5	38.	hxg5	a4						
19.	Rxe5	g6	39.	bxa4	Rc4		1:0				
20.	Qf2	Nd7	40.	a5	bxa5						

Figure 23: A game from the 1963 Botvinnik versus Petrosian match.

The first column, #G, is the number of games currently analyzed to produce the adversary pattern knowledge base. The #Ch and #M columns are the number of geometric chunks and textual opening move chunks respectively. The first number in the #M column represents the total number of chunks and the second number is the chunks pertaining to white moves, for TEST1, or black moves, for TEST2. The next three columns are the number of predictions made by IAM (#P), the number of predictions which exactly matched the adversary's ensuing move (#C), and the

```
                    TEST2--Botvinnik playing Black.

  1.  d4      Nf6        21.  Rhg1    Kh7       41.  Ra1     Kg7
  2.  c4      g6         22.  Nb5     Rf7       42.  Ra6     Rb7
  3.  Nc3     d5         23.  Nd4     Re8       43.  Ra8     Kf6
  4.  Qb3     dxc4       24.  Nf3     Bh6       44.  Rc8     Ne5
  5.  Qxc4    Bg7        25.  Ng5+    Bxg5      45.  Ke3     Nd7
  6.  e4      O-O        26.  Rxg5    Nc4       46.  Rc6+    Kf7
  7.  Be2     Nc6        27.  Rdg1    Rg8       47.  e5      Nf8
  8.  Nf3     Nd7        28.  Kc2     b6        48.  Rf6+    Kg7
  9.  Be3     Nb6        29.  b3      Nd6       49.  Ke4     b5
 10.  Qc5     Bg4        30.  f3      Rd7       50.  Rc6     Kf7
 11.  d5      Nd7        31.  R5g2    Rdd8      51.  Rxc5    Ne6
 12.  Qa3     Bxf3       32.  a4      Nf7       52.  Rd5     Ke7
 13.  Bxf3    Nd4        33.  Bc1     e5        53.  Be3     Rb8
 14.  O-O-O   Nxf3       34.  Be3     exf4      54.  Rd6     b4
 15.  gxf3    Nb6        35.  Bxf4    Rd7       55.  Ra6     Rb5
 16.  Qb3     Qd7        36.  Rd2     Rxd2+     56.  Ra7+    Ke8
 17.  h4      h5         37.  Kxd2    Rd8+      57.  f4      Kf8
 18.  f4      e6         38.  Ke2     c5        58.  f5
 19.  dxe6    Qxe6       39.  a5      Rd7
 20.  Qxe6    fxe6       40.  axb6    axb6                  1:0
```

Figure 24: Another game from the 1963 Botvinnik versus Petrosian match.

number of predictions which identified the correct piece to move, but selected the wrong square on the board as the destination (CPWL). The final three columns reveal the statistical measures for the number of analogous positions found (%G), the accuracy of the predictions that were made (%C), and the percentage of predictions identifying the correct piece to be moved (%C+P). The percentage of correct predictions for the entire game can be obtained by multiplying the appropriate column (%C or %C+P) by the percentage of analogous positions. We will use the percentage of correct predictions and correct piece identifications for the entire game in our analysis of IAM's performance.

Table 1: IAM Performance Measures for TEST1

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	28	24/10	31	5	8	54.3	16.1	41.9
22	53	31/14	47	6	7	82.4	12.7	27.6
36	66	38/16	46	6	9	80.7	13.0	32.6
45	68	39/16	38	5	3	66.7	13.1	21.0
80	86	52/22	36	6	4	63.1	16.6	27.8

The games analyzed by IAM in Table 1 are for the adversary Botvinnik and are from the following tournaments:

- #G = 12, games are from the 1948 Hague-Moscow Tournament.

- 22, new games are from the 1951 Bronstein Match.

- 36, new games are from the 1954 Smyslov Match.

- 45, new games are from the 1957 Smyslov Match.

- 80, new games are from the 1958 Smyslov Match and the 1960 and 1961 Tal Matches.

The sixth and ninth columns of Table 1 are significant in demonstrating that the adversary model is focusing on the correct area of the game board only using knowledge of the pattern chunks known by an adversary.

IAM's analysis of the first twelve games played by Botvinnik produces correct predictions of adversary moves for nine percent of the total moves and correctly

identifies the piece to be moved in twenty-three percent of the total moves in the game. The size of the geometric chunk knowledge base nearly doubles with IAM's analysis of the next ten games played by Botvinnik. We can see from the second row of the table that the correct piece is still identified for twenty-three percent of the total moves. However, the percentage of correct predictions increases to almost eleven percent of the total game moves. The new chunks enable the adversary model to convert one of the correct piece identifications into a correct prediction.

IAM's induction based learning technique continues to improve the performance of the adversary model following the next fourteen games that were analyzed. The third row of Table 1 shows the correct predictions remaining constant, but the percentage of correct piece identifications is now over twenty-six percent of the total moves.

We also notice that the number of predictions or analogous positions which showed a fifty percent increase between the first and second rows of the table starts to drop in the third row. This curtailment in the quantity of predictions made by the adversary model continues throughout the rest of the table. Our inference engine tries to maintain existing chunks on the game board. The reduction in analogous positions is due to the newest chunks in the knowledge base already existing in the location of an earlier chunk that was only partially completed. Because the new chunk exists in its entirety, the analogy which produced the previous chunk's predictions is discarded by the inference engine making the predictions.

The last two rows of Table 1 demonstrate an interesting problem revealed in our experiments. First the correct prediction percentage fluctuates back to nine percent before regaining its optimal eleven percent rate in the last row. The percentage of correct piece identifications for both rows drops below twenty percent. The cause of this apparent decline in performance is similar to the cause for the reduced prediction rates described above. The presence of newly learned chunks on the game board prohibits the suggestion of moves for completing other chunks that require borrowing a piece from the existing chunk. Competition between the chunks for the pieces on the game board reduces the total number of predictions made by the adversary model and likewise the number of correct piece identifications.

To understand chunk competition, we must first realize that once a chunk is in the knowledge base it will always identify the same board configurations of a particular game as being analogous. The chunk will then suggest the appropriate move or moves to maneuver the missing piece into the correct location to complete the chunk. Pseudo-code which suggests the appropriate move to complete a chunk is shown in Figure 25. As new chunks are learned and added to the knowledge base, the likelihood that the missing piece required to complete the earlier chunk is contained in a new chunk increases. If a new chunk which also contains the missing piece of the earlier chunk exists on the board and owns the specific piece required to complete the earlier chunk, then the earlier chunk's move suggestion is rejected by the inference engine due to the existing chunk.

The pseudo-code in Figure 25 requires that the board position (i, j) correspond to the upper-leftmost square of the defined chunk. First we determine the piece that is required to complete the chunk and then we search the board representation to locate the required piece. If the piece can be moved from its current square to the square which completes the chunk without violating the movement rules of chess, then the corresponding move is suggested. An individual

```
If number of pieces needed to complete the chunk C > 1
Then see if either a king-side or queen-side castle will complete C
Else Begin
        For A = 1 to the width of C do  ;; the chunk to be completed
        For B = 1 to the length of C do
            If C_{A,B} < > 0 AND X_{i+A,j+B} < > C_{A,B}
            Then Begin
                    P ← C_{A,B}
                    ;; P is the piece needed to complete chunk C
                    a ← A
                    b ← B
                End Begin
        For I = 1 to 8 do
            For J = 1 to 8 do
                If X_{I,J} = P
                Then If piece P can be moved from board location (I,J)
                            to location (i+a, j+b)
                        Then suggest this move as a possible next move
        End Begin
```

Figure 25: Pseudo-code to suggest possible adversary moves.

chunk can produce multiple suggestions if more than one of the required piece to complete the chunk exists on the board and can be moved to the proper location.

IAM's accuracy with respect to the entire game is fairly consistent, with correct predictions ranging between nine and eleven percent of the total moves of the game. A slightly wider range, fourteen to twenty-three percent of the total moves, is obtained for predictions identifying the correct piece to be moved. While being able to correctly predict approximately ten percent of an adversary's moves is an extraordinary accomplishment, the usefulness of our methodology for coaching existing chess programs is dependent on the reliability of the predictions or the ratio of the correct predictions to the total number of predictions made for a game.

4.2.1.2 Reducing incorrect predictions through likelihood

The percentage of correct predictions to total predictions made by IAM, shown in Table 1, falls between twelve and seventeen percent. This means that a chess program's evaluation function can rely on IAM's predictions one sixth of the time. In Chapter 3, we noted that IAM calculates a probability or likelihood which, after being modified by the heuristic inference rules, is used to determine the move that the adversary will make next.

By disregarding predictions with low likelihood values, we improve the performance ratio of correct prediction to total predictions. The likelihood values and corresponding prediction ratios and percentages for the results shown in Table 1 are displayed in Table 2. The top row of Table 2 displays the likelihood values used to select predictions. Entries in the table accrue the predictions from the next

highest likelihood value so that the second column for likelihood values less than twenty-five percent are identical to the total prediction ratios shown in Table 1. The percentage of correct predictions to total predictions is shown beneath each ratio pair.

We can see that by limiting the predictions to be considered by the existing chess program to have a likelihood value of greater than fifty percent, the predictions of IAM are from seventy-five to one hundred percent correct. This enables a chess program to use IAM's knowledge without fear of blundering into the trap of utilizing an incorrect prediction. The only problem which arises from requiring predictions to have a high likelihood is that correct predictions with smaller likelihoods are ignored. From Table 2 we see that the percentage of game moves which are correctly predicted drops to just over five percent.

Table 2: Effect of Likelihood on Prediction Ratios

Games	< 25%	25-30%	30-40%	40-50%	> 50%
12	5 / 31 16.1%	4 / 9 44.4%	3 / 6 50 %	3 / 5 60 %	3 / 4 75 %
22	6 / 47 12.7%	3 / 7 42.8%	3 / 6 50 %	3 / 4 75 %	3 / 3 100 %
80	6 / 36 16.6%	3 / 6 50 %	3 / 3 100 %	3 / 3 100 %	3 / 3 100 %

4.2.1.3 Performance versus the adversary as the black player

The second part of our initial experiment to determine the base performance levels of IAM uses TEST2 to establish IAM's capabilities against an adversary playing the black pieces. TEST2 is the game from Botvinnik's championship match

against Petrosian which followed the TEST1 game. The length of the TEST2 game is 58 turns. The performance results for IAM during the simulated TEST2 game are displayed in Table 3. The meanings of each of the headings in Table 3 are the same as for Table 1. Because the knowledge acquisition and inductive learning segments have already been completed the first three columns of Table 3 are identical to Table 1, except the textual opening move knowledge base numbers represent the total chunks and the chunks for the black opening moves of the adversary.

Table 3: IAM Performance Measures for TEST2

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	28	24/14	17	0	4	29.3	0.0	23.5
22	53	31/17	17	1	3	29.3	5.9	23.5
36	66	38/22	17	1	4	29.3	5.9	29.4
45	68	39/23	17	1	4	29.3	5.9	29.4
80	86	52/30	23	1	5	39.7	4.3	26.0

We immediately see that Table 3 demonstrates learning or an increase in performance as more games are analyzed by IAM. The correct predictions rise from zero to a consistent two percent (5.9 percent multiplied by 29.3 percent) of the total moves of the game. Identification of the piece to be moved increases gradually from seven percent to over ten percent of the game moves.

We previously mentioned a modification to the textual chunk knowledge base to capture the responsive nature of black's play. The results of this modification are presented in Table 4. The differences between Table 3 and Table 4, other than the

size of the textual chunk knowledge base, are the results of the opening move predictions which rely on the textual move knowledge base.

The modification to the textual chunk knowledge base, which simulates the responsive nature of human black piece play, increases the prediction capabilities of IAM. The last two rows of Table 4 show a one hundred to two hundred percent increase in correct predictions from the original model. To verify this result, we randomly selected another game from the Petrosian match in which Botvinnik played the black pieces. Again, the original adversary model made one or two correct predictions for each size of the knowledge bases. However, the adversary model with the modified textual knowledge base produced an additional correct prediction, effectively doubling the performance characteristics of the adversary model.

Table 4: Performance with Responsive Modification

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	28	32/22	17	0	4	29.3	0.0	23.5
22	53	42/28	17	1	3	29.3	5.9	23.5
36	66	57/41	17	1	3	29.3	5.9	23.5
45	68	64/48	17	3	4	29.3	17.6	41.2
80	86	89/67	23	2	4	39.7	8.7	26.1

Even with the modification to account for the responsive nature of black play, IAM's performance against an adversary playing the black pieces is only fifty percent of the performance against an adversary playing white. The reason for this is that the quality of knowledge available about the adversary as a black player is inferior to the knowledge available about the adversary as a white player. The first several

knowledge bases, produced from the twelve and twenty-two game analyses, have a maximum of twenty-three percent of the games played as black, or five of twenty-two games. The larger knowledge bases have a greater quantity of black games and a more even distribution with forty-six percent of the eighty game knowledge base being black games.

The quality of the knowledge about the adversary as a black player is still inferior. Sixty-seven percent of the white games played by the adversary resulted in won games. Only forty percent of the black games resulted in a winning outcome. Remember that we normally use a series of lost games to indicate an imminent change in the adversary's strategy or playing style and chunks which are only found in lost games have their move suggestion probabilities reduced by the prediction inference engine. Therefore, the poor quality of information available about the adversary as a black player is directly responsible for the reduced performance of the adversary model against the future games of the adversary in which the adversary plays the black pieces.

4.2.2 Performance with Respect to Time

We conducted a two part experiment to reveal temporal dependencies of the adversary model. In the first part of our experiment, the geometric and textual chunk knowledge base of the adversary model is created from observations of the past performances of an adversary which occur in time near to the TEST1 game. This is to detect any reliance on the recency of knowledge.

Botvinnik's 1961 chess match versus Tal is used to analyze games that are recent to the 1963 TEST1 game. IAM's performance is shown in Table 5. The overall correct predictions are fewer, only seven percent of total game moves, and the identification of the correct piece to be moved is comparable to the forty-five game knowledge base. The only noticeable change produced by analyzing recent games is that the number of analogous situations detected by the adversary model is lower than any of the previous knowledge bases using older knowledge. The reduction in predictions corresponds to an increase in the correct prediction to total prediction ratio, thus increasing the reliability of IAM's predictions.

Table 5: Effect of Recency on IAM Performance for TEST1

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	16	26/13	16	4	4	28.0	25.0	50.0
15	29	26/13	14	4	4	24.5	28.5	57.1

The second part of the temporal experiment tries to simulate the effect of forgetting on an adversary's pattern knowledge by aging the geometric pattern chunks. Once a knowledge base exists in the adversary model for a particular adversary, the geometric pattern chunks have their number of observances halved prior to incrementally adding new chunks or reinforcement of existing chunks from analyzing additional games played by the adversary. If a chunk's number of observations is less than one, then the chunk is deleted from the knowledge base. This has the effect of removing chunks learned from earlier/older games if they are not repeated in more recent games.

The performance of IAM against TEST1 with aging geometric chunks is shown in Table 6. The textual chunks column is identical to Table 1, since textual chunks are not being aged and therefore has been replaced by a column, REM, indicating the number of chunks removed from the knowledge base due to aging. The 1961 Tal games of Botvinnik were not used in this experiment hence, the last row indicates only sixty-five games were analyzed to produce the adversary model used by IAM.

Table 6: Effect of Aging/Forgetting on IAM's Performance

#G	#Ch	REM	#P	#C	CPWL	%G	%C	%C+P
12	28	N/A	31	5	8	54.3	16.1	41.9
22	53	0	47	6	7	82.4	12.7	27.6
36	48	18	36	4	4	63.2	11.1	22.2
45	32	25	11	3	0	19.2	27.2	27.2
65	53	20	22	2	3	38.5	9.0	22.7

We see from Table 6 that some of the geometric chunks are consistently repeated and therefore not deleted from the knowledge base. The geometric patterns of pieces that are surviving in the adversary model are primarily chunks consisting of pawn structures.

The introduction of forgetting via an aging mechanism for the geometric chunks produces a steadily decreasing overall performance of IAM against TEST1. While we do not propose that chess masters never forget, a better factor for removing knowledge from the knowledge base would involve a combination of the age of a specific piece of knowledge and a size limitation on the number of chunks

which can be stored in the knowledge base. In Chapter 2, research indicated that chess masters store 10,000 to 100,000 patterns. The size of the adversary model knowledge base after analyzing eighty games is one hundred and seventy-five patterns, half of which are geometric chunks and the other half are textual patterns. Many thousands of games, similar to the life experiences of a chess master, would need to be analyzed to produce a knowledge base size approaching 10,000 patterns.

4.2.3 Performance with Respect to Textual Chunk Size

The results for IAM already shown have all been for a textual move pattern knowledge base which learns the statistical patterns associated with the first five opening moves. Other textual move patterns are learned, but are not used by IAM for predicting the opening move sequence of the adversary. We modify the textual pattern learning algorithm to learn and analyze the first ten moves and the first fifteen moves of every game. The performance results of IAM operating with the ten opening moves textual pattern learning is presented in Table 7 and the fifteen opening moves knowledge base results are presented in Table 8. Each table shows the knowledge base obtained from analyzing Botvinnik's three earliest tournaments. We feel that the results are sufficient to demonstrate the effect of changing the opening move knowledge base size.

IAM's performance with the larger textual knowledge bases is better for the case of analyzing twelve games, with a twenty percent increase in correct predictions and a fifteen to twenty-three percent increase in correct piece to be moved identification, for the ten and fifteen move knowledge bases respectively. However,

Table 7: Performance (TEST1) with 10 Opening Moves

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	28	60/31	32	6	9	56.1	18.7	46.8
22	53	84/46	48	6	7	84.2	12.5	27.0
36	66	112/57	49	6	8	85.9	12.2	28.5

Table 8: Performance (TEST1) with 15 Opening Moves

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	28	102/52	34	6	10	59.6	17.6	47.0
22	53	152/81	50	6	6	87.7	12.0	24.0
36	66	213/110	51	6	7	89.5	11.7	25.4

the performance returns to the five opening moves textual knowledge base level for the adversary model produced by analyzing twenty-two games and actually drops below the five move knowledge base performance level for the thirty-six game adversary model.

The increase in performance for the twelve game adversarial model textual knowledge base results from predictions covering the sixth through tenth or fifteenth moves which were not found to contain positions analogous to the geometric chunks. The responsibility for prediction shifts from the geometric chunks to the textual chunks. The return to normal performance levels and the drop in performance experienced for the twenty-two and thirty-six game adversary models occurs from the subsequent geometric chunks which are learned and able to find analogous positions during the extended opening time period. The newly learned geometric chunks are

filling in the gaps for the sixth and later moves that are present in the smaller twelve game adversary model.

The number of textual patterns stored in the adversary model's knowledge base increases significantly with the change to capture additional opening moves as textual patterns. For the ten opening moves textual pattern modification the textual portion of the knowledge base is tripled and the fifteen move modification exceeds quintupling the number of textual chunks. Since no apparent benefit is produced by the extension of the textual patterns, we do not recommend this modification except when the number of geometric chunks is small (e.g., less than fifty chunks).

4.2.4 Performance with Respect to Geometric Chunk Size

Our next experiment is to demonstrate the adequacy of the four-by-four or sixteen square geometric chunk size. As shown in Appendix B, the majority of the chunks induced by IAM could have been acquired with a three-by-three or nine square chunk size, but we use the geometric chunks to save the ending piece count and this requires a sixteen element chunk. It is possible that since we discard chunks which exceed the maximum size, larger chunks which will improve the adversary model's performance may be lost.

We research the performance of the adversary model with respect to geometric chunk size by altering the chunk acquisition algorithm following the convolution operator to accept chunks of a twenty-five square area and again to accept chunks of a thirty-six square area. The results for the first three adversary models are shown in Table 9, for the five-by-five area chunks, and Table 10, for the

six-by-six area chunks. The 'nc' in the textual chunks column indicates 'no change' from the Table 1 values.

Table 9: IAM Performance (TEST1) with Five-by-Five Chunks

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	28	nc	31	5	7	54.3	16.1	38.7
22	54	nc	46	6	6	80.7	13.0	26.0
36	67	nc	45	6	6	78.9	13.3	26.6

Table 10: IAM Performance (TEST1) with Six-by-Six Chunks

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	29	nc	31	5	7	54.3	16.1	38.7
22	55	nc	45	6	6	78.9	13.3	26.6
36	68	nc	45	6	6	78.9	13.3	26.6

The number of new chunks induced can be obtained by subtracting the number of geometric chunks found by the four-by-four algorithm shown in Table 1 from the number of chunks shown in the above tables. One five-by-five geometric chunk and one six-by-six geometric chunk are produced for the thirty-six game analysis of row three in the tables. The five-by-five chunk is shown in Figure 26 along with another chunk that is induced for both the five-by-five and four-by-four chunk sizes.

The performance is identical for the twelve game adversary model regardless of the geometric chunk size. We see in Figure 26, that the smaller chunk is contained within the larger chunk and is capable of making all predictions involving
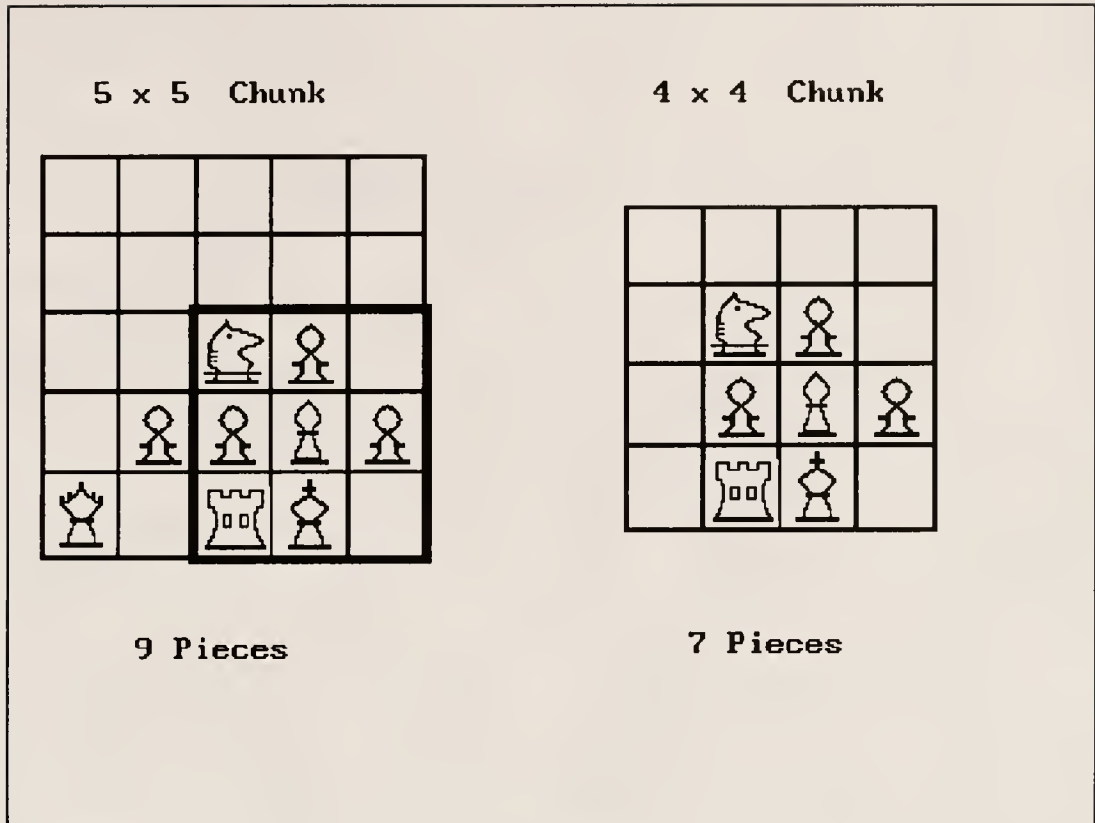
Figure 26: Chunk acquired with 5x5 size and corresponding smaller chunk.

the seven pieces of the chunk. However, the five-by-five chunk requires that an additional two pieces, the queen and extra pawn, be in place to use this chunk for predicting the identical moves of the four-by-four chunk. IAM's performance actually suffers for the next two larger adversary models due to interference from the five-by-five chunk induced during the twenty-two game analysis.

The new chunks, which cover a larger board area, already exist on the game board during a time that the smaller chunks identify analogous positions. The concurrent existence of the new larger chunk causes the prediction inference engine to prohibit the smaller chunk move suggestions, some of which were able to predict the correct piece to be moved. Because IAM's inference engine simulates the

hierarchical construction of complex chunks from smaller less complex chunks, the overall performance of IAM is better when we use the smallest chunk size of four-by-four. Based on IAM's performance we define a primitive chunk to contain three to nine pieces and cover a board area of no more than sixteen squares.

4.2.5 The Effect of Domain Specific Knowledge

Our adversary modeling methodology uses only minimal amounts of domain knowledge, such as the movement capabilities of each piece, to predict the actions of an adversary. Current chess programs use a myriad of domain specific knowledge to evaluate board positions while selecting moves. Our final experiment is to determine the effect on performance when domain specific knowledge is contained in the adversary model.

We augment the heuristic rules used by the prediction inference engine to include knowledge concerning pieces being en prise. The purpose of the heuristic is to invalidate move suggestions when a major piece is en prise, unless the suggested move will change the board location of the en prise piece. Our heuristics are simply rejecting inferior move choices and not attempting to suggest any alternate moves that will save the en prise piece.

IAM's performance for the twelve game and twenty-two game adversary models against TEST1 is shown in Table 11. The performance for the adversary model created from the analysis of twelve of Botvinnik's games is identical to the original results given in Table 1. However, the performance for the twenty-two game adversary model shows a reduction in the number of predictions made by the

adversary model from forty-seven to thirty-four. This corresponds to an increase in the reliability of the predictions and further increases the utility of the adversary model's predictions for existing chess programs, as discussed in section 4.2.2.

Table 11: Effect of Domain Specific Knowledge

#G	#Ch	#M	#P	#C	CPWL	%G	%C	%C+P
12	28	24/10	31	5	8	54.3	16.1	41.9
22	53	31/14	34	6	6	74.5	17.6	35.3

The addition of the small amount of knowledge concerning en prise pieces increases the reliability of IAM's correct move predictions by five percent and is higher than all of the reliability values displayed in Table 1. An explicit example of IAM's performance improvement is seen in the chunks' predictions for the position in Figure 22. Since the queen is en prise, only the fourth chunk's suggestion which predicts the correct move is permitted by the new heuristics, thus increasing the correct predictions for that game by one. IAM's improved performance with the addition of a minuscule amount of domain specific knowledge indicates that further improvements in reliability can be expected with the continued addition of greater amounts of domain specific knowledge.

4.2.6 Repeatability of the Demonstrated Results

The performance results given in Section 4.2, have been obtained from the analysis of TEST1 and TEST2 by IAM. To verify these results and gain confidence in the statistical significance of the displayed results, we had IAM predict adversary moves in five additional games. Two of these games are from the end of the

Hague-Moscow Tournament; therefore, only the games preceding these test games were used to create the adversary model knowledge base and the other three are additional games from the 1963 Petrosian match.

IAM's performance in all five of the additional games is very similar to the results which have already been reported. The two white games had durations of forty-eight and sixty-three moves with IAM making five and six correct predictions respectively, or approximately ten percent of the total moves, and identification of the piece to be moved approximately twenty percent of the total game moves. Performance in each of the three additional black games is similar to the demonstrated results with correct predictions ranging from one to three per game and slightly higher correct piece identification.

For the total seven games, IAM's performance is consistent across the three games in which the adversary played the white pieces and the four games in which the adversary played the black pieces. The ability of IAM to repeatedly obtain similar performance levels increases the statistical significance of IAM's performance results.

CHAPTER 5
RESULTS

We have successfully acquired geometric and textual chunks that have been repeated by an adversary in more than one game. Our assumption that these chunks form the foundations of an adversary's evaluation criteria for the domain is validated by the actual predictions made by the adversary model. Approximately ten percent of all game moves, when the adversary plays the white pieces, are correctly predicted. Furthermore, for over twenty percent of the moves in a game we can identify the piece which is about to be moved.

The identification of which piece is to be moved indicates that our adversary model is focussing attention in the same locale as the adversary's attention. Additional support for the similarities between the adversary modeling methodology's perception of the board and the adversary's board evaluation is found by looking at the two moves following the adversary model's predictions. For the TEST1 game, within two moves of a prediction made by IAM, the predicted move was executed once and the piece which IAM predicted to be moved was used four times. Although IAM's predictions were a little premature, we can see that the adversary model is cognizant of the area of the board where action is about to take place.

The performance of the adversary model is exceptional when we account for the quality of the information analyzed. Table 12 shows the win to loss ratio for the

Botvinnik games analyzed by IAM. High quality information is obtained from won games and lost games provide us with questionable-quality knowledge. Again, we see that IAM's lack of performance against the adversary as the black player results from sixty percent of the analyzed black games having ended in a loss.

Table 12: Ratio of Won and Lost Games for White and Black

	White Games	Black Games
Total Games Analyzed	43	37
Won Games	29	15
Lost Games	14	22

Our research has resulted in three specific advancements for intelligent programs operating in adversarial domains. Through use of the adversary modeling methodology we can reduce opening book sizes while maintaining comparable performance capabilities, reveal critical paths in the game tree not previously considered, and reduce search complexity by pruning unnecessary nodes from the game tree. The advancements are detailed individually with respect to their application in the chess domain.

5.1 Reducing Opening Book Size

Most current chess programs make use of opening books ranging in size from 5,000 to 60,000 bytes (Newborn & Kopec, 1990). The size of an opening book affects the speed of the program's play with larger books taking more time to process. Research is being performed to reduce the size of opening books for chess programs

by taking advantage of redundancies among various opening lines (White, 1990). White's method affords a thirty-eight percent savings in space, reducing a 5,000 byte opening book to 1900 bytes.

The collection of textual chunks referring to opening moves simulates a human master's analysis of an opponent and requires substantially less space than standard opening books. If we assume the same memory constraints used by White of two bytes per move, then the opening moves learned by the adversary model require 312 bytes. This is the maximum size for the opening book from Table 4, using the responsive code for black moves by the adversary after analyzing eighty games. We also need three additional bytes per move to save the number of times a move has been observed, the color of pieces played by the adversary for each move, and the mean occurrence time of each move, increasing our storage requirements to 468 bytes. Even with our additional storage requirements, the adversary model's textual knowledge base provides over a ninety percent savings in the space requirements for an opening book.

To utilize the adversary model, a chess program should analyze the collected opening sequences saved in the textual knowledge base to identify opening sequences that are unfamiliar to the adversary. Using our definition of five game turns for the standard duration of a known opening, we can still store fourteen complete opening sequences, with moves for both players, and maintain twice the space savings of previous opening book reduction methods. This will enable chess programs to gain a strategic advantage over their opponents by using unfamiliar openings while still reducing their opening book size by eighty percent.

5.2 Revealing New Search Paths in the Game Tree

Adversarial domains, especially games, frequently use tree models to represent important information in the planning process. Domains using tree representations must use a search algorithm for selecting the optimal solution from the tree. Adversarial domain tree representations are very large, 10^{43} nodes for chess, which prohibits an exhaustive search to find the solution. Heuristics are used to prune branches from the tree with only partial knowledge of what is contained down the pruned branches and to evaluate current board positions to determine the end result of a path even though a leaf node has not yet been reached by the search algorithm.

We improve the performance of search algorithms by revealing branches or paths that would not normally be considered. In Figure 27, we present a subtree from the larger game tree for a hypothetical chess game. The computer chess program at the MAX node level is attempting to select the optimal move from node A through use of the standard minimax search with alpha-beta pruning. The evaluation function values for each of the nodes at the bottom of the search range from negative five, for a sure loss to the computer, through positive five, for a sure win to the computer, with zero representing a drawn outcome.

The left subtree returns a value of zero to the adversary's MIN level decision node. The MAX player, the computer, now knows that a draw can be obtained by selecting the move corresponding to the left branch of this game tree. Likewise, the right branch returns a value of negative one, or a slight chance of a loss, as the MIN player's movement choice. The first node evaluated for the center branch has a
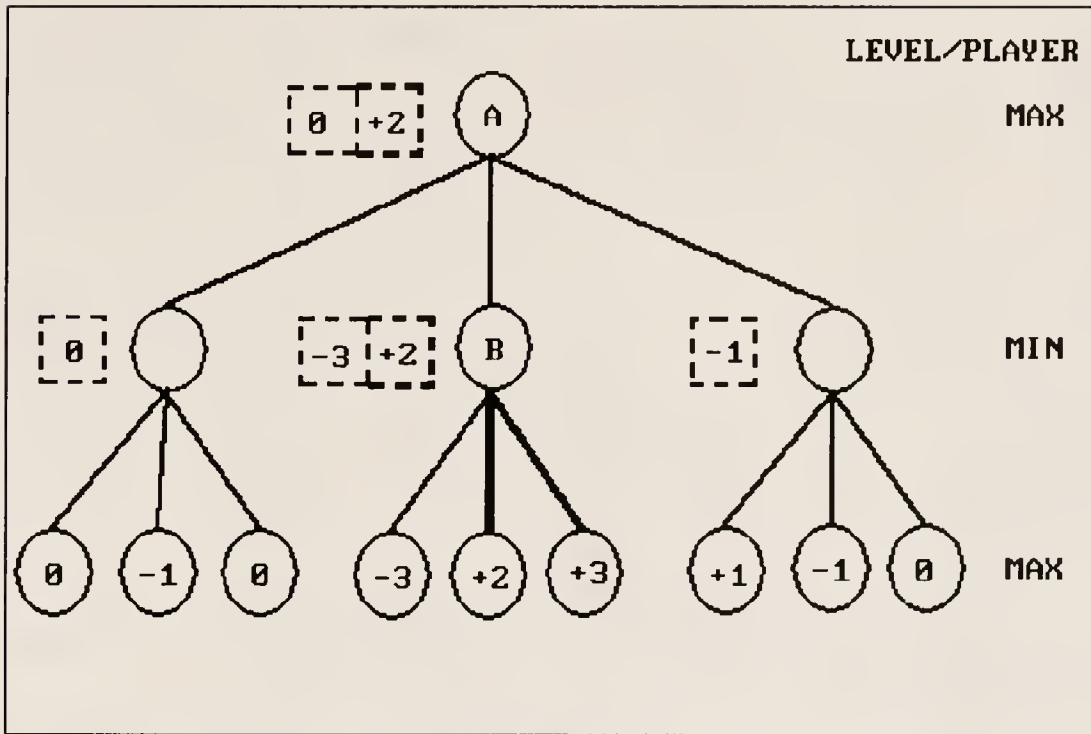
Figure 27: Hypothetical game/search tree.

negative three value. Alpha-beta pruning would cut the remaining two branches

from node B because a better value is already available to the MAX player from the

right subtree. Therefore, negative three becomes the value associated with selecting

the center branch of the tree. The minimax algorithm will choose the maximum

node value from the MIN level from the right subtree, resulting in a probable draw

as the outcome of the game.

With the appropriate knowledge in the adversary model, a much better

outcome could be obtained by the chess program. The move that is represented by

the center branch of the game tree extending from node A corresponds to the

proffering of a gambit by the MAX player and the right-most branch from node B

corresponds to the acceptance of the gambit by the MIN player. The adversary

model of this adversary, the MIN player, shows that the adversary has never accepted a gambit in previous play. Therefore, the negative three valued node should not be considered as a viable move choice for the adversary to consider. The new game tree, minus the gambit acceptance node, now returns a value of positive two to the B node as MIN's choice. The new best move for the MAX player using knowledge of the adversary's past performances is to choose the center branch of the tree and proffer the gambit.

A specific example of the ability of our adversary model to reveal adversary move choices that were not previously considered by a chess programs search algorithm comes from the game shown in Figure 19 and using the geometric chunks displayed in Figure 6. Figure 28 shows the board position prior to the twenty-first move to be made by Botvinnik and the chunk used in predicting the move. Two commercial chess programs were asked to analyze this position and select the next move to be made.

The SARGON 4 program, searching to a depth of six ply, selected the Rc1 move and the GNUCHESS program, searching to a depth of eight ply, selected the Ne5 move for the position shown in Figure 28. IAM predicted that Botvinnik would move the knight located at d3 to either the b4 or e5 square. Both move choices were predicted with equal likelihood since each move will create the same chunk. Each of the commercial programs, which have a program option to reveal their search trees, never considered the Nb4 move in their respective searches. Botvinnik actually made the Nb4 move choice that was one of the two IAM predictions. This example
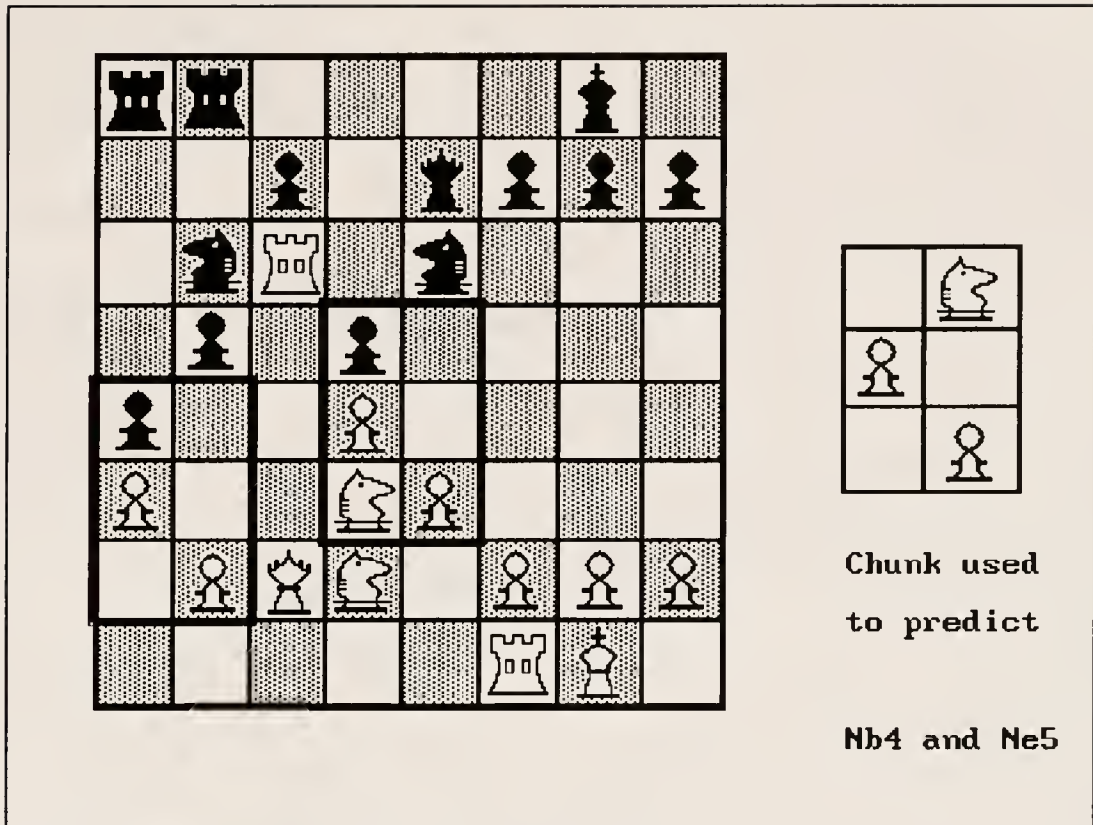
Figure 28: Prediction of Botvinnik's next move.

illustrates that the adversary model is capable of alerting existing chess programs to move choices that are either not considered at all, as above, or not considered to be optimal selections.

5.3 Reducing Search Complexity Through Automatic Pruning

Another improvement to artificial intelligence search strategies performance is made by the adversary model through eliminating the need to perform search at certain levels of a game tree. Providing a unique descending branch from a node has the same effect as pruning all of the other branches from that node, which reduces the search complexity.

As we have shown in Chapter 4, IAM is capable of predicting the exact move to be made by an adversary ten percent of the time. For an additional fifteen percent of the game, IAM predicts the correct piece to be moved by the adversary. The contributions of this knowledge in the adversarial domain of football are obvious. If the coach of one team knows the exact play to be called by the adversary every tenth down and which player will end up with the ball on one and one half of the remaining downs, then the coach has a significant strategic and tactical advantage over the opposing team.

For adversarial domains using tree representations of domain knowledge, the knowledge acquired by the adversary model succeeds in reducing the tree complexity which corresponds to a deeper search for the same number of evaluated tree nodes. When a search is performed to a depth of twenty ply with a branching factor of N, the maximum number of branches which descend from all nodes of the tree, then with a prediction rate of ten percent we will be able to accurately predict the individual branch which will be followed by an adversary from one of the nodes. This effectively prunes N minus one branches from the tree at that node and all of the associated sub-branches descending from the pruned nodes. By adding a single branch to the tree, we can now search to a depth that is one greater than the original search. A comparison of the two searches with N equal to four is demonstrated graphically in Figure 29, which shows the standard search tree and the extended tree obtained by using the adversary model's knowledge for the same number of expanded nodes.
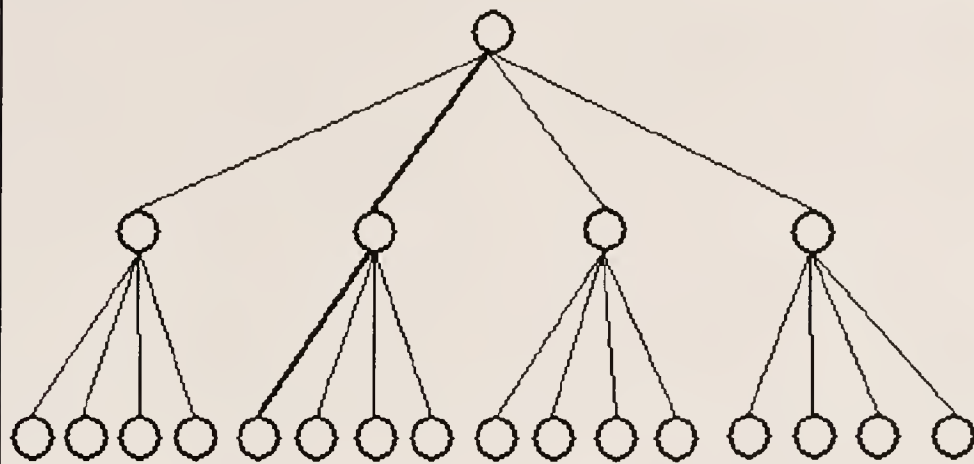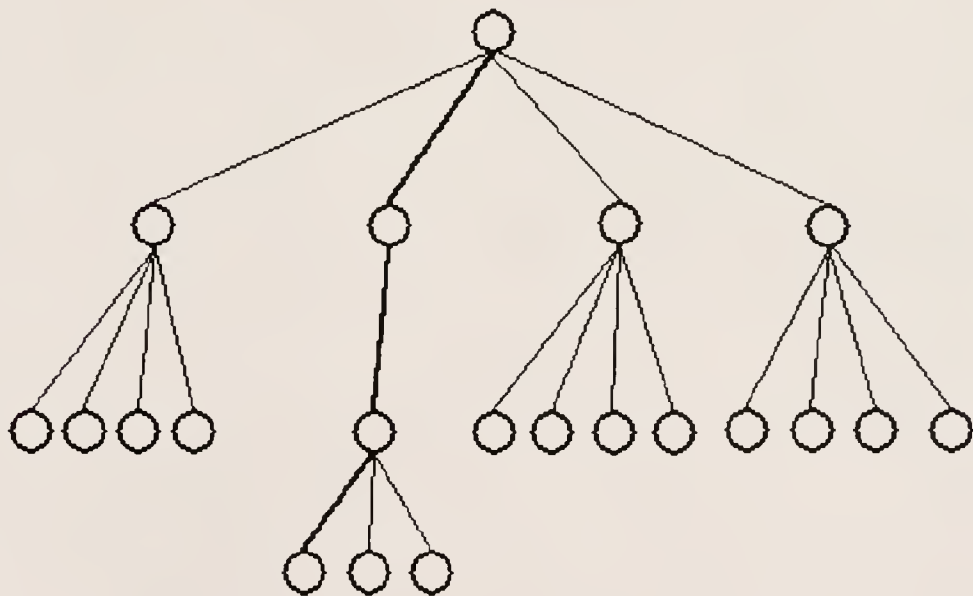
Figure 29: Standard search tree and the IAM improved search tree.

The time order for the standard search algorithm is

$$O(\text{standard search}) = k1 * N^m = O(N^m),$$

where 'm' is the depth of the search and 'k1' is a constant greater than one corresponding to the time required to actually perform the search and evaluation algorithms. With 'p' representing the percentage of correct predictions made by the adversary model and the constant 'k2' corresponding to the time required to use the adversary model's knowledge base and prediction algorithm, the new time order of the search algorithm is

$$O(\text{prediction search}) = k2 + (k1 * N^{m * (1 - p)}) = O(N^{m * (1 - p)}).$$

This shows that the time order can be reduced, since 'p' is less than or equal to one, by an exponential factor with the addition of a linear constant to the time equation.

Essentially, a search algorithm can now afford to search (m * p) extra ply deeper in the game tree with only a linear increase in time. In Chapter 4, we mentioned that search algorithms would require reliable predictions. The mathematical equations above assume perfect reliability. We have seen that a one hundred percent reliability can be obtained by only considering chunks from the adversary model knowledge base that have a fifty percent of greater likelihood value. The use of high likelihoods correspondingly decreases the percentage of correct predictions from ten percent to five percent of the total game moves. With a five percent prediction rate, a tree search of forty ply will have its time order reduced from $O(N^{40})$ to $O(N^{(40 * (1 - (0.05/2)))})$ or $O(N^{39})$. A forty-one ply depth search can be accomplished in the same time as the previous forty ply search. The reason for

dividing the 'p' prediction percentage value by two is that only every other branch in the game tree is predictable by the adversary model.

Because we require perfect reliability and we are using induction so that only domain situations that are analogous to situations which have been induced will produce predictions, the effect of gaining one ply of search every twenty game turns is a reasonable result. In section 4.2.5, we showed that the inclusion of even a minimal amount of domain specific knowledge in the adversary model significantly decreases the number of wrong predictions that are forecast by the model. With greater amounts of domain specific knowledge, we believe that the ten percent correct prediction performance rate can be obtained with nearly perfect reliability hence, doubling the exponential savings to the time order of adversarial domain search algorithms.

We now assume that the addition of a larger quantity of domain specific knowledge will enable the adversary model to make only accurate predictions or predictions that identify the correct piece to be moved. Even greater time savings are obtained if we use the predictions which identify the correct piece to be moved. Utilizing the prediction heuristics of correct pieces, in addition to correct moves, requires that the search algorithm still performs a search at each depth level of the game tree. The improved search is still faster than the standard search algorithm, since only those moves that are going to move the predicted piece need to be examined. For example, in a search tree which has a breadth of search 'N', of five, only two or three of the suggested moves will actually use the piece predicted by the

adversary model. An example of the improved search tree from using the adversary model's correct piece identifications is shown in Figure 30.
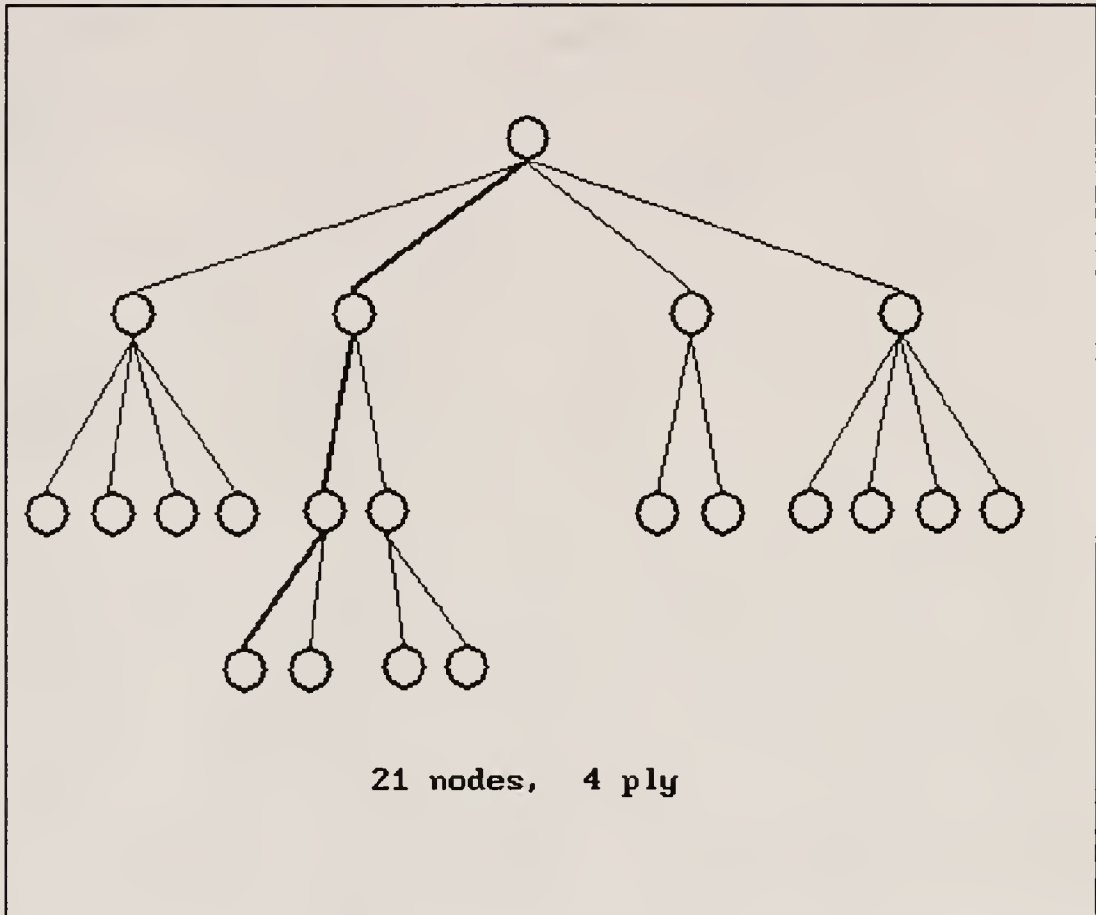

Figure 30: Search tree obtained from Correct Piece identification.

The time order equation uses 'q' to represent the percentage of actual moves for which the adversary model can predict either the correct move or identify the correct piece to be move. We must remember from Chapter 4, that 'q' is twice as large as 'p'. The new reduced branching factor for nodes where predictions are made is 'R' and we can guarantee that 'R' is less than 'N'. The new time order is

$$O(\text{improved prediction search}) = (k2 * R^{m \cdot q}) + (k1 * N^{m \cdot (1-q)})$$

$$= O(N^{m \cdot (1-q)}) \text{ or } O(R^{m \cdot q}).$$

The actual time order depends on the value of 'q'. Because of the need to actually perform a reduced search at each of the tree nodes associated with adversary model predictions, the resultant additive factor is also an exponential instead of a linear constant. If 'q' is less than one half, or fifty percent, then the order is the first alternative. Otherwise, the second alternative is larger and will become the time order of the search algorithm. With the twenty percent correct piece identification performance demonstrated by IAM in Chapter 4, the improved algorithm reduces the time order of the standard search algorithm from $O(N^{20})$ to $O(N^{18})$ or a two ply increase in the search capabilities of the standard search algorithm. This is one order of magnitude better than the prediction based search which uses only the correct move predictions, although the reduction in complexity is not as intuitively appealing.

5.4 Summary of Research Contributions

In Chapter 2, background research claimed that the speed and depth of search are critical factors in the playing ability of intelligent game programs which use tree representations of the game domain. We have reduced the size requirements of opening books for chess programs with a corresponding reduction in the amount of time required to use the smaller opening book. The predictions of an adversary's move by the adversary model has resulted in an exponential savings in time for standard game search algorithms. Time savings realized from the use of the adversary modeling methodology correspond to increases in the depth of search that

is accomplished by a search algorithm for the same amount of time. Deeper searches in the game tree provide better tactical solutions to chess problems.

The textual chunks which are used to reduce the opening book size have a correlative strategic contribution to adversary domain programs. A chess program can use the knowledge in the textual portion of the adversary model's knowledge base to select opening sequences of play that have not been experienced by the adversary. This produces a strategic benefit to the chess program by placing the adversary in unfamiliar territory.

Improvements in playing ability of current adversarial programs are realized through the revelation of previously unconsidered search paths. The adversary model enables a chess program to determine the most likely move to be made by an adversary based on the adversary's past performance.

As increases in search speed through technological advancements in hardware capabilities approach their physical limitations, further increases in intelligent program performance must be accomplished by using knowledge. The adversary model provides a methodology for simulating the cognitive preparations that experts in adversarial domains use when preparing to meet an adversary. Previous methods have limited themselves to small stages of the game. We have produced a generic method which can be used to augment intelligent program performance at all stages of a game.

CHAPTER 6
CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

We have developed and implemented an adversary modeling methodology which acquires the cognitive chunks that are used by an adversary to evaluate complex domain situations. The chunks are acquired by employing a technique frequently used by adversarial domain experts which is to analyze the previous performances of a specific adversary. Chunks are comprised of both textual and visual patterns that are repeated at different times in the adversarial domain. For chess, this means a pattern that occurs in two or more separate games.

Our performance results indicate that we can accurately predict ten percent of an adversary's moves when high quality information, or a high percentage of won games, has been analyzed. Additionally, we can predict the piece to be moved for over twenty percent of the adversary's move choices in a game. These results reveal that adversarial domain experts frequently use basic cognitive techniques that are domain independent to choose their moves. The cognitive technique that we rely on is the simplification of complex domain situations to situations that have been encountered previously.

Current chess program capabilities result from a combination of search speed and use of domain specific knowledge. Our adversary model is not exempt from the use of domain specific, since we have seen in Chapter 4 that the addition of domain specific knowledge increases the reliability of the adversary model's predictions. We have demonstrated that by using our domain independent knowledge the complexity and time order of a search algorithm can be reduced. The use of our adversary modeling methodology's domain independent knowledge increases the performance capabilities of current chess programs. We believe that the continued increase in performance of intelligent chess programs requires the use of additional knowledge including domain independent knowledge similar to an adversary model.

6.2 Future Research Directions

We have tried to indicate, throughout this dissertation, meaningful extensions to the adversary modeling methodology's current research. A summary of these extensions and some additional thoughts for future research directions are presented below.

6.2.1 Acquire More Geometric Chunks

We have seen that the performance of the adversary model increases with the addition of more chunks as long as the chunks do not interfere with each other. The convolution algorithm identifies all pieces that are horizontally or vertically proximal or can defend another piece diagonally. Chunks which exceed a sixteen square size are ignored. Due to the hierarchical composition of complex chunks from smaller

simpler chunks, we feel that chunks which are originally to big to be evaluated by the induction algorithm can be broken down into smaller chunks that maintain the meaning of the original chunk. Trying to eliminate pieces from a chunk by ignoring all pieces beyond a particular horizontal or vertical axis of the game board may destroy the strategic and tactical significance of the chunk. A better method would be to devise heuristic rules which could eliminate fringe pieces from a chunk while maintaining the strategic and tactical importance of the new chunk's pieces.

6.2.2 Increase the Knowledge Contained in Each Chunk

The knight is currently included in chunks without regard to its movement capabilities. The convolution algorithm can be modified to use a five-by-five template when a knight is contained in a chunk to capture the movement capabilities of the knight. Increasing the template size, with suitable modifications for each individual piece's movement capabilities as shown in Figure 31, enables the adversary modeling methodology to acquire chunks that are meaningful, but not proximal. The original templates, shown in Figure 8, used by the convolution algorithm returns a bitmap of values that are independent of the center piece. To save space since the new bitmap values will need to store values up to 2^{56}, the new templates are dependent on the central piece and further processing by the chunk acquisition algorithm will need to refer back to the central piece at each board location. If the enlarged templates tried to use the general piece independent approach of the original templates then bitmap values up to 2^{255}, or one bit in each bitmap for every position of the fifteen by fifteen templates of the bishop, rook, and queen.
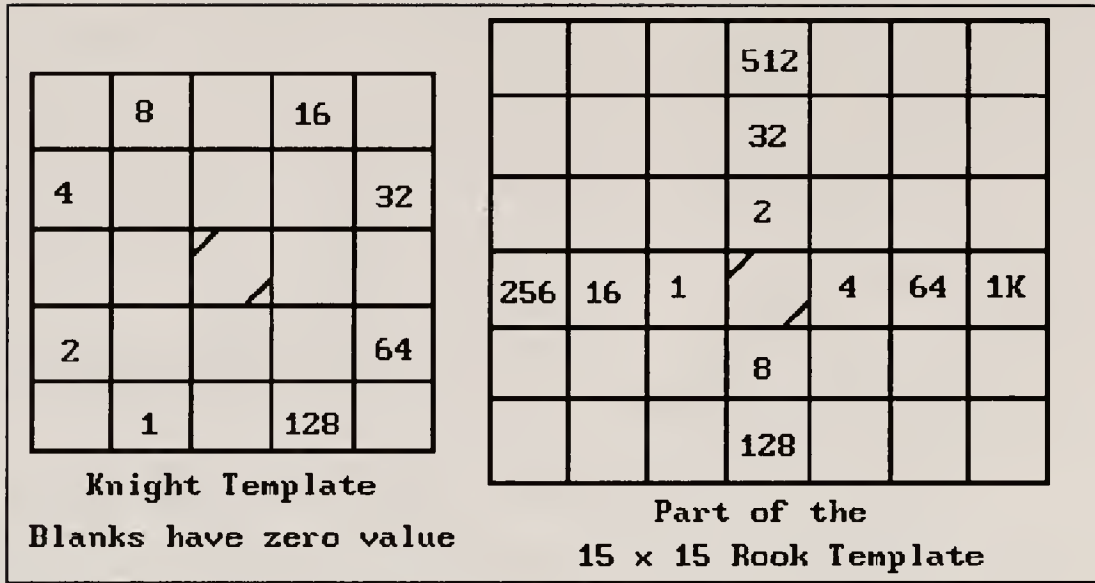
Figure 31: Knight template and partial Rook template.

Additional knowledge can be obtained by dropping our similarity constraint of chunks containing one color of pieces. Permitting chunks to contain both colors of pieces simultaneously has already been used to acquire knowledge about the pawn structures and pawn chains found in an adversary's games. Along with the increased template size mentioned above, chunks with both color pieces enables the adversary modeling methodology to acquire knowledge concerning the use of forks, pins, and skewers. In addition to the vision based convolution method for acquiring chunks, another computer vision technique lends itself to the acquisition of non-proximal relationships between pieces. The Hough transform is used to identify the beams of light which pass through a specific point in an image. While Hough transforms are cumbersome to implement in vision domains, the discrete nature of board games would permit an efficient implementation of the Hough transform which would identify pieces that both lay on a particular line of the board and would thus capture

distance relationships without the need for over-sized templates in the convolution operator.

Storing the knowledge acquired by either a Hough transform or the convolution operator with fifteen by fifteen templates would require a sixty-four square chunks size, or quadruple the current chunk size. Although we can implement the sixty-four square chunks, the additional space required would make the adversary model inefficient and less desirable as a slower coach. The use of semantic networks would enable the adversary modeling methodology to store all chunks in a space efficient manner. Semantic networks would require a change in the functional description of our implementation of the adversary modeling methodology, but would not change the theoretical implications of the methodology.

6.2.3 Place More Adversarial Knowledge in the Adversary Model

We have already seen in Chapter 4, the inclusion of domain specific knowledge increases the reliability of the adversary model's predictions. Increasing the domain specific knowledge available to the adversary model will further augment the capabilities of the adversary modeling methodology. For game domains which already have a great amount of domain specific knowledge in their evaluation functions, cooperation between the adversary model and the evaluation function might serve to eliminate inferior predictions. However, many adversarial domains do not have a large amount of domain specific information contained in their application programs. We can easily modify the prediction inference engine heuristic rules to increase the quantity of domain specific knowledge available to the adversary model.

Other types of global knowledge are also available to increase the adversary model's capabilities. In Chapter 2, we mentioned that cultural and educational differences affect the strategic and tactical planning of adversaries. Kotov and Yudovich (1961) state that large scale instruction and training in chess is backed by the Soviet government via the Trade Unions. The sociological and cultural traits of the idealized Soviet man are the primary traits of the chess school. The desire to win corresponds to ardent patriotism. Specific strengths and weaknesses are displayed by similar cultural groups. A statistical analysis of the 1973 international chess tournaments revealed several statistically significant chess weaknesses produced by a Soviet chess education (Mednis, 1978). Table 13 displays some of the weaknesses that were found by Mednis's analysis. The global knowledge about Soviet chess players displayed in Table 13 would a chess program which knew that its opponent was a Soviet player. A chess program playing against a Soviet opponent could use the fourth row of the table to realize that time pressure is relevant in one fourth of Soviet chess losses and correspondingly a chess program could reduce its search depth from a possible high of thirteen ply to a maximum of ten ply to force the Soviet player into a time sensitive situation.

Our current research focuses on the perceptual evaluation of domain situations by a domain expert. The next step performed by chess masters after evaluating the board position is to plan their attack or defense. The prediction capabilities of the adversary model correspond to the evaluation mechanisms used by the adversary. Suggesting moves which are strategically and tactically superior

against a specific adversary is the next logical step. The adversary model already infers the general playing style of an adversary and acquires the opening sequences displayed by an adversary. These two pieces of knowledge can be used to form plans against the opening and playing styles of the adversary. The inclusion of the global cultural knowledge and domain specific knowledge would enable the adversary model to suggest move sequences that would be strategically advantageous. The ability to make move suggestions would have a further impact on search complexity since both the MIN, which is already affected, and the MAX node levels would be coached by the adversary model.

Table 13: Statistical Analysis of Soviet Chess Weaknesses.

STRATEGIC KNOWLEDGE	GAMES LOST FROM EFFECT OF KNOWLEDGE	PERCENTAGE
Lose more with Black.	39 of 68	57%
Weak middle-game.	44 of 68	65%
More errors made from defending than attacking positions.	52 of 68	76%
Suffer from time pressure.	18 of 68	26%
More errors due to strategy than tactics.	41 of 66	62%

Our adversary modeling methodology successfully predicts many of the moves that an adversary will make. The predictions are based on domain independent knowledge and demonstrate that the use of domain independent knowledge as well as domain dependent knowledge are the critical elements in continuing to improve adversarial domain application program performance. The current model focuses

primarily on the cognitive and perceptual processes used by an adversary to evaluate the domain. The textual and visual chunks, which form the foundations of the evaluation mechanisms employed by experts, are the domain independent knowledge that enables the adversary model's predictions. By further increasing the knowledge contained in the adversary model with domain dependent and global knowledge, the adversary model will be able to suggest strategically advantageous moves as well as predicting likely adversary moves. The use of the adversary model as a coach will greatly enhance the performance capabilities of existing adversarial domain planning programs.

APPENDIX

The following glossary of chess and artificial intelligence terms is meant to assist the reader in understanding the examples and explanations of this dissertation.

alpha-beta pruning:

A method of reducing the complexity of search trees which utilize minimax processing. The general idea is that once a value has been established at the MIN level of a search tree, then subsequent values of other subtrees to be backed up to the current MIN level which are less than the established value will force all other branches of that subtree to be pruned, or alpha pruned. Likewise, once a value has been established for a MAX level of the tree then any subsequent subtrees which have a value greater than the current MAX level value will force the remaining branches of that subtree to be beta pruned.

algebraic notation:

A method for describing the moves of a chess game and the position of the pieces. Each column of the chess board is designated by a letter from 'a' to 'h' and each row is designated by a number from 1 to 8, as shown below.

a8	b8	c8	d8	e8	f8	g8	h8
a7	b7	c7	d7	e7	f7	g7	h7
a6	b6	c6	d6	e6	f6	g6	h6
a5	b5	c5	d5	e5	f5	g5	h5
a4	b4	c4	d4	e4	f4	g4	h4
a3	b3	c3	d3	e3	f3	g3	h3
a2	b2	c2	d2	e2	f2	g2	h2
a1	b1	c1	d1	e1	f1	g1	h1

Moves are described by giving one of the letters N, B, R, Q, or K to designate the piece that is being moved (no letter translates to a pawn move) followed by the letter and number of the destination square. For example, Nf3 describes the move of a knight to the king's-bishop-three square. The one row is always white's home row and correspondingly the eight row is always black's home row.

en prise:

A piece is "en prise" if it can be captured by the opponent on the next move.

evaluation function:

Evaluation functions are used by chess games to determine the value, or goodness, of a specific board position. Typical evaluation functions use domain dependent tactical and positional knowledge, such as material balance and control of the center squares of the game board, in determining a position's value.

fork:

When a knight has the positional ability to capture two different pieces.

heuristic:

A "rule of thumb" which will produce a solution to a problem. However, the solution produced by the heuristic is not guaranteed to be optimal.

major piece:

Any chess piece other than a pawn. A king, queen, rook, bishop, or knight piece.

material balance:

A number which represents the difference between the number and quality of pieces currently on the board for each side. Each piece has its own value with the value of the stronger pieces being greater than the value of the weaker pieces, such as the value nine for a queen and the value three for a knight. The sum of the value for each piece remaining on the board for each side is obtained. Then the opponent's material sum is subtracted from your material sum to obtain the material balance.

minimax:

An algorithm that decides the optimal tree path to follow by alternately backing up the minimum and maximum of all leaf nodes. The MAX player tries to obtain the maximum possible value of all branches.

pawn structure: Any group of pawns of the same color which are either diagonally or horizontally adjacent to each other.

pin: When a queen or rook is attacking a piece which lies between the queen or rook and another piece of greater value than the intervening piece.

ply: A ply refers to a single level of a game tree or one half of a game turn. Each game turn consists of one move by each of the players involved. A ply is therefore, the move of one of the two players.

quiescent: A board position in which no captures can take place and the king is not in check. Quiescence is also used to refer to board positions in which a capture can occur, but no subsequent captures are possible.

skewer: Similar to a pin except the threatening piece is attacking along a diagonal (i.e. a bishop).

REFERENCES

Adelson-Velsky, G. M., Arlazarov, V. L., & Donsky, M. V. (1975). Some Methods of Controlling the Tree Search in Chess Programs. Artificial Intelligence, 6(4), 361-371.

Adelson-Velsky, G. M., Arlazarov, V. L., & Donsky, M. V. (1988). Algorithms for Games. Springer-Verlag, New York.

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1974). The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, MA.

Aliston, W., & Weiskopf, D. (1984). The Complete Baseball Handbook. Allyn and Bacon, Boston.

Anderson, J. R. (1980). Cognitive Psychology and Its Implications. W. H. Freeman, New York.

Anderson, J. R. (1987). Causal Analysis and Inductive Learning. In Proceedings of the Fourth International Workshop on Machine Learning, Irvine, CA, 288-299.

Andriole, S. J., Black, H. H., & Hopple, G. W. (1986). Intelligent Aids for Tactical Planning. IEEE Transactions on Systems, Man, and Cybernetics, SMC-16(6), 854-864.

Angluin, D., & Smith, C. H. (1983). Inductive Inference: Theory and Methods. Computing Surveys, 15(3), 237-269.

Barr, A., & Feigenbaum, E. A. (Eds.). (1981). The Handbook of Artificial Intelligence Volume 1. Addison-Wesley, Reading, MA.

Bell, A. G. (1978). The Machine Plays Chess? Pergamon Press, New York.

Berliner, H. (1979). On Construction of Evaluation Functions for Large Domains. In Proceedings Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan, 53-55.

Berliner, H. (1988). HITECH Becomes First Computer Senior Master. AI Magazine, 9(3), 85-87.

Berliner, H., & Ebeling, C. (1988). <u>Pattern Knowledge and Search: The SUPREM Architecture</u>. Technical Report CMU-CS-88-109, Carnegie-Mellon University, Pittsburgh.

Berliner, H., & Goetsch, G. (1984). <u>A Quantitative Study of Search Methods and the Effect of Constraint Satisfaction</u>. Technical Report CMU-CS-84-147, Carnegie-Mellon University, Pittsburgh.

Blum, L., & M. Blum, M. (1975). Toward a Mathematical Theory of Inductive Inference. <u>Information and Control</u>, <u>28(2)</u>, 125-155.

Bolc, L. (Ed.). (1987). <u>Computational Models of Learning</u>. Springer-Verlag, New York.

Bonasso, R. P. (1988). What AI Can Do for Battle Management. <u>AI Magazine</u>, <u>9(3)</u>, 77-83.

Bond, M. H. (Ed.). (1986). <u>The Psychology of the Chinese People</u>. Oxford University Press, Oxford, UK.

Boring, E. G. (Ed.). (1945). <u>Psychology for the Armed Services</u>. The Infantry Journal, Washington, DC.

Bratko, I., & Michie, D. (1980). An Advice Program for a Complex Chess Programming Task. <u>The Computer Journal</u>, <u>23(4)</u>, 353-359.

Bratko, I., Tancig, P., & Tancig, S. (1986). Detection of Positional Patterns in Chess. In <u>Advances in Computer Chess 4</u>, 113-126. Pergamon Press, New York.

Campbell, M. S. (1988). <u>Chunking as an Abstraction Mechanism</u>. PhD thesis, Carnegie-Mellon University, Pittsburgh.

Candland, D. K. (1980). Speaking Words and Doing Deeds. <u>American Psychologist</u>, <u>35(2)</u>, 191-198.

Carbonell, J. G. (1981). Counterplanning: A Strategy-Based Model of Adversary Planning in Real-World Situations. <u>Artificial Intelligence</u>, <u>16(3)</u>, 295-329.

Case J., & Smith, C. (1983). Comparison of Identification Criteria for Machine Inductive Inference. <u>Theoretical Computer Science</u>, <u>25(2)</u>, 193-220.

Chase, W. G., & Simon, H. A. (1973). Perception in Chess. <u>Cognitive Psychology</u>, <u>4(1)</u>, 55-81.

Chase, W. G., & Simon, H. A. (1988). The Mind's Eye in Chess. In <u>Readings in Cognitive Science</u>, 461-494. Morgan Kaufmann, San Mateo, CA.

Christensen, J., & Korf, R. E. (1986). A Unified Theory of Heuristic Evaluation Functions and Its Application to Learning. In <u>Proceedings AAAI-86 The Fifth National Conference on Artificial Intelligence</u>, Philadelphia, 148-152.

Cimbala, S. J. (1987). <u>Artificial Intelligence and National Security</u>. Lexington Books, Lexington, MA.

Crookall, D., Greenblat, C. S., Coote, A., Klabbers, J., Watson, D. R. (Eds.). (1987). <u>Simulation-Gaming in the Late 1980's</u>. Pergamon Press, New York.

Daley, R. P., & Smith, C. H. (1986). On the Complexity of Inductive Inference. <u>Information and Control</u>, <u>69(1-3)</u>, 12-40.

Davis, B. (1956). <u>Gray Fox Robert E. Lee and the Civil War</u>. Fairfax Press, Lehigh Valley, PA.

Davis, P. K. (1988a). <u>Applying Artificial Intelligence Techniques to Strategic-Level Gaming and Simulation</u>. Technical Report N-2752-RC, Rand Strategy Assessment Center, Santa Monica, CA.

Davis, P. K. (1988b). A new analytic technique for the study of deterrence, escalation control, and war termination. <u>Simulation</u>, <u>50(5)</u>, 195-202.

Davis, P. K., Bankes, S. C., & Kahan, J. P. (1986). <u>A New Methodology for Modeling National Command Level Decisionmaking in War Games and Simulations</u>. Technical Report R-3290-NA, Rand Strategy Assessment Center, Santa Monica, CA.

DeJong, K. A., & Schultz, A. C. (1988). Using Experience-Based Learning in Game Playing. In <u>Proceedings of the Fifth International Conference on Machine Learning</u>, Ann Arbor, MI, 284-290.

Department of the Army (1986). <u>Field Manual 7-7J The Mechanized Infantry Platoon and Squad (Bradley)</u>. Washington, DC.

Dunnigan, J. F. (1982). <u>How To Make War</u>. William Morrow, New York.

Ebeling, C. (1986). <u>All the Right Moves: A VLSI Architecture for Chess</u>. PhD thesis, Carnegie-Mellon University, Pittsburgh.

Elithorn, A., & Banerji, R. (Eds.). (1984). Artificial and Human Intelligence. Elsevier, New York.

Erickson, M. D., & Zytkow, J. M. (1988). Utilizing Experience for Improving the Tactical Manager. In Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI, 444-450.

Ermarth, F. W. (1978). Contrasts in American and Soviet Strategic Thought. International Security, 3(2), 138-155.

Evans, L. (1970). Chess Catechism. Simon and Schuster, New York.

Findler, N. V., & Meltzer, B. (1971). Artificial Intelligence and Heuristic Programming. Elsevier, New York.

Frey, P. W. (1983). Chess Skill in Man and Machine. Springer-Verlag, New York.

Geitner, P. (1989). Computer is Pawn to Chess Master. The Gainesville Sun, 114(109), 1A,6A.

Gelo, J. H. (1988). Chess World Championships All the Games 1834-1984. McFarland and Company, Jefferson, NC.

Gruner, W. P. (1988). Intelligence Information for Submarines. The Submarine Review, (Jan), 21-28.

Hamilton, J. (1990). Kasparov's Visit to Boston. Chess Life, 45(1), 30-31,92.

Holding, D. H. (1985). The Psychology of Chess Skill. Lawrence Erlbaum Associates, Hillsdale, NJ.

Horgan, D. D., Millis, K., & Neimeyer, R. A. (1989). Cognitive Reorganization and the Development of Chess Expertise. International Journal of Personal Construct Psychology, 2, 15-36.

Horowitz, A. (1973). The World Chess Championship A History. Macmillan, New York.

Kodratoff, Y., & Michalski, R. S. (Eds.). (1990). Machine Learning An Artificial Intelligence Approach Volume III. Morgan Kaufmann, San Mateo, CA.

Kopec, D., & Newborn, M. (1987). Belle and Mephisto Dallas Capture Computer Chess Titles at FJCC. Communications of the ACM, 30(7), 640-645.

Kotov, A., & Yudovich, M. (1961). The Soviet School of Chess. Dover Publications, New York.

Laird,J., Rosenbloom, P., & Newell, A. (1986). Universal Subgoaling and Chunking. Kluwer Academic, Norwell, MA.

Langer, W. C. (1972). The Mind of Adolf Hitler. Basic Books, New York.

Langley, P. (1985). Learning to Search: From Weak Methods to Domain Specific Heuristics. Cognitive Science, 9(2), 217-260.

Lasker, E. (1973). Modern Chess Strategy. David McKay Company, New York.

Lee, K., & Mahajan, S. (1988). A Pattern Classification Approach to Evaluation Function Learning. Artificial Intelligence, 36(1), 1-25.

Lenat, D. B. (1982). The Nature of Heuristics. Artificial Intelligence, 19(2), 189-249.

Lenat, D. B., & Feigenbaum, E. A. (1987). On the Thresholds of Knowledge. Technical Report AI-126-87, Microelectronics and Computer Technology Corporation (MCC), Austin, TX.

Lenat, D. B., Hayes-Roth, F., & Klahr, P. (1979). Cognitive Economy in Artificial Intelligence Systems. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan, 531-536.

Levy, D. (1984). The Joy of Computer Chess. Prentice Hall, Englewood Cliffs, NJ.

Levy, D. (Ed.). (1988). Computer Games I. Springer-Verlag, New York.

McMillen, D. H. (Ed.). (1984). Asian Perspectives on International Security. St. Martin's Press, New York.

Mednis, E. (1978). How to Beat the Russians. David McKay Company, New York.

Mednis, E. (1990). Move Orders in the Opening: The Modern Master's Tool. Chess Life, 45(6), 14-16.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.). (1983). Machine Learning An Artificial Intelligence Approach. Morgan Kaufmann, San Mateo, CA.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.). (1986). Machine Learning An Artificial Intelligence Approach Volume II. Morgan Kaufmann, San Mateo, CA.

Miller, G. A. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review, 63(2), 81-97.

Mitchell, T. M., Carbonell, J. G., & Michalski, R. S. (Eds.). (1986). Machine Learning A Guide to Current Research. Kluwer Academic, Norwell, MA.

Muggleton, S. (1990). Inductive Acquisition of Expert Knowledge. Addison-Wesley, Reading, MA.

Narendra, K., & Thathachar, M. (1989). Learning Automata: An Introduction. Prentice Hall, Englewood Cliffs, NJ.

Newborn, M., & Kopec, D. (1989). Results of the Nineteenth ACM North American Computer Chess Championship. Communications of the ACM, 32(10), 1225-1230.

Newborn, M., & Kopec, D. (1990). The Twentieth Annual ACM North American Computer Chess Championship. Communications of the ACM, 33(7), 92-104.

Nunn, J., & Griffiths, P. (1987). Secrets of Grandmaster Play. Macmillan, New York.

Pachman, L. (1975). Complete Chess Strategy: First Principles of the Middle Game. Doubleday, New York.

Polya, G. (1973). How To Solve It. Princeton University Press, Princeton, NJ.

Pomerantz, J. R. (1986). Visual Form Perception: An Overview. In Pattern Recognition by Humans and Machines Volume 2 Visual Perception, 1-30. Academic Press, Orlando, FL.

Pritchard, D. B. (1973). Go: A Guide to the Game. Faber and Faber, Winchester, MA.

Reiss Games (1974). Go: The Ancient Oriental Game. New York.

Riggins, J., & Winter, J. (1984). Gameplan The Language and Strategy of Pro Football. Santa Barbara Press, Santa Barbara, CA.

Ritter, G. X., Wilson, J. N., & Davidson, J. L. (1988). Image Algebra: An Overview. Technical Report TR-88-05, University of Florida, Gainesville, FL.

Robbins, J. (1988). America's Red Army. The New York Times Magazine, (Apr), 38-46.

Rosch, E. H. (1973). Natural Categories. Cognitive Psychology, 4(3), 328-350.

Rosenbloom, P. S. (1982). A World-Championship-Level Othello Program. Artificial Intelligence, 19(3), 279-320.

Roycroft, A. J. (1988). Expert Against Oracle. In Machine Intelligence 11, 347-373. Oxford University Press, Oxford, UK.

Ryan, P. J. (1988). Some Commandments of Submarine Warfare. The Submarine Review, (Jan), 60-63.

Saariluoma, P. (1984). Coding Problem Spaces in Chess A Psychological Study. Societas Scientiarum Fennica, Helsinki, Finland.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 3(3), 210-229.

Samuel, A. L. (1967). Some Studies in Machine Learning Using the Game of Checkers II - Recent Progress. IBM Journal of Research and Development, 11(6), 601-617.

Schonberg, H. C. (1973). Grandmasters of Chess. J. B. Lippincott Company, Philadelphia.

Shannon, C. E. (1950). Programming a Computer for Playing Chess. The Philosophical Magazine, 41(314), 256-275.

Shapiro, A. D. (1987). Structured Induction in Expert Systems. Addison-Wesley, Reading, MA.

Simon, H. A., & Gilmartin, K. (1973). A Simulation of Memory for Chess Positions. Cognitive Psychology, 5(1), 29-46.

Sleeman, D., & Brown, J. S. (1982). Intelligent Tutoring Systems. Academic Press, Orlando, FL.

Snow, W. P. (1867). Lee and His Generals. Richardson and Company, New York.

Sokolovskiy, V. D. (1968). Soviet Military Strategy. Macdonald and Jane's, Stanford, CA.

Soltis, A. (1976). Pawn Structure Chess. David McKay Company, New York.

Tadepalli, P. (1989). Knowledge Based Planning in Games. Technical Report CMU-TR-89-135, Carnegie-Mellon University, Pittsburgh.

Utgoff, P. E. (1986). Machine Learning of Inductive Bias. Kluwer Academic, Norwell, MA.

Utgoff, P. E., & Saxena, S. (1987). Learning a Preference Predicate. In Proceedings of the Fourth International Workshop on Machine Learning, Irvine, CA, 115-121.

Von Neumann, J., & Morgenstern, O. (1953). Theory of Games and Economic Behavior. Princeton University Press, Princeton, NJ.

Wallace, J. (1990). Bloody Chosin: The Blind Lead the Brave. U. S. News & World Report, 108(25), 37-43.

Waterman, D. A. (1970). Generalization Learning Techniques for Automating the Learning of Heuristics. Artificial Intelligence, 1(1-2), 121-170.

White, J. F. (1990). The Amateurs' Book Opening Routine. ICCA Journal, 13(1), 22-26.

Wilcox, B. (1985). Reflections on Building Two Go Programs. SIGART Newsletter, (94), 29-43.

Wilkins, D. E. (1980). Using Patterns and Plans in Chess. Artificial Intelligence, 14, 165-203.

Wilkins, D. E. (1982). Using Knowledge to Control Tree Searching. Artificial Intelligence, 18(1), 1-51.

Witkin, G. (1990). New Drug Warriors: Lasers, Labs, and Coke-Eating Bugs. U. S. News & World Report, 108(7), 20-21.

Woolf, B. P. (1984). Context Dependent Planning In A Machine Tutor. PhD thesis, University of Massachusetts, Amherst, MA.
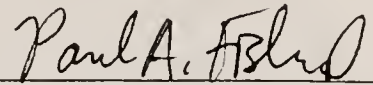
Young, P. R., & Lehner, P. E. (1986). Applications of a Theory of Automated Adversarial Planning to Command and Control. <u>IEEE Transactions on Systems, Man, and Cybernetics</u>, <u>SMC-16(6)</u>, 806-812.

Zagare, F. C. (1984). <u>Game Theory Concepts and Applications</u>. Sage Publications, Newbury Park, CA.
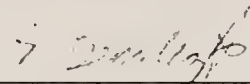
BIOGRAPHICAL SKETCH

Steven Michael Walczak received his Bachelor of Science degree with a major field of study in mathematics from the Pennsylvania State University in 1981. Following his undergraduate education, Mr. Walczak was employed by the Department of Defense where he was responsible for the design and implementation of various software projects including a signal analysis expert system. While working for the government, he acquired his Master of Science in computer science from the Johns Hopkins University. In 1986, Mr. Walczak moved to Florida and began working for two different government contractors. His primary responsibilities were the development of an automated knowledge acquisition system, the development of an intelligent tutoring system, and the design of a message forwarding system. Mr. Walczak returned to academia in 1988 as a full-time student working for his Doctor of Philosophy degree in computer and information sciences from the University of Florida.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.
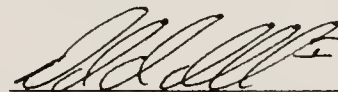
Paul A Fishwick, Chairman
Assistant Professor of Computer and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Manuel E. Bermudez
Associate Professor of Computer and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.
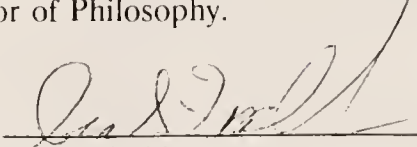
Douglas D. Dankel II
Assistant Professor of Computer and
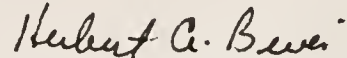Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Walter R. Cunningham
Professor of Psychology

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Ira S. Fischler
Professor of Psychology

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.
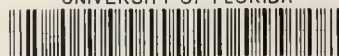
December 1990

Winfred M. Phillips
Dean, College of Engineering

Madelyn M. Lockhart
Dean, Graduate School